QURAN MEMORIZATION PLAN GENERATION USING ARTIFICIAL INTELLIGENCE

AHMAD OSAMA KENTAR

MOHAMMAD YAHYA BATHEEB

ABDULKAREEM YAHYA ALMAHLAWI

DEPARTMENT OF COMPUTER SCIENCE

KING ABDULAZIZ UNIVERSITY

DECEMBER 2024

QURAN MEMORIZATION PLAN GENERATION USING ARTIFICIAL INTELLIGENCE

AHMAD OSAMA KENTAR

MOHAMMAD YAHYA BATHEEB

ABDULKAREEM YAHYA ALMAHLAWI

THIS REPORT IS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE BACHELOR DEGREE IN COMPUTER SCIENCE

DR. ABDULLAH MARISH ALI

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF COMPUTERS INFORMATION TECHNOLOGY

KING ABDULAZIZ UNIVERSITY

DECEMBER 2024

Declaration of Originality

We hereby declare that this project report is based on our original work except for citations and quotations, which had been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at KAU or other institutions.

Student Name	ID	Signature	Date
Ahmad Osama Kentar	2148843		December 2024
Mohammad Yahya Batheeb	2148844		December 2024
Abdulkareem Yahya Almahlawi	2148850		December 2024

Abstract

This project presents an AI-driven solution to enhance Quran memorization through personalized and adaptive learning plans, addressing the limitations of traditional fixed-schedule methods. Traditional approaches often fail to accommodate individual differences in memorization speed, retention capacity, and availability, leading to student disengagement or inefficiency. Leveraging machine learning techniques—Long Short-Term Memory (LSTM) networks and Reinforcement Learning (RL)—the proposed system dynamically tailors memorization and revision plans based on real-time student performance and historical data. LSTM models analyze sequential learning patterns to predict baseline memorization goals, while RL adjusts these goals in response to recitation feedback, optimizing workload balance between new memorization and revision. The methodology integrates a robust database architecture (PostgreSQL) to manage student profiles, Quranic content (surahs and verses), and recitation sessions, ensuring seamless data flow between AI models and user interfaces. Collaborating with a Quranic education company provided access to structured historical data, enabling the training of LSTM models on realistic memorization patterns. Testing revealed moderate predictive accuracy, with LSTM explaining 47%, 70%, and 64% of variance in new memorization, minor revision, and major revision tasks, respectively. However, higher error rates in major revisions (1,265 letters) highlight opportunities for refinement. The backend, built with Python, TensorFlow, and SQLAlchemy, supports real-time adjustments, while the Bootstrap-based frontend offers educators intuitive tools for tracking progress and updating plans. Key outcomes include a responsive system that adapts to individual learning trajectories, enhances retention through spaced repetition, and provides educators with actionable insights. Future directions emphasize deploying the system as a scalable web/mobile application, integrating automated voice recognition for recitation evaluation, and expanding user roles to include administrators and parents. By bridging AI capabilities with pedagogical needs, this project demonstrates the transformative potential of technology in Quranic education, offering a flexible, data-driven framework to empower learners globally.

المستخلص

يقدم هذا المشروع حلاً مدعومًا بالذكاء الاصطناعي لتعزيز حفظ القرآن عبر خطط تعليمية مُكَيَّفة وشخصية، لمعالجة قيود الأساليب التقليدية ذات الجداول الثابتة. غالبًا ما تفشل المناهج التقليدية في مراعاة الاختلافات الفردية في سرعة الحفظ، وقدرة الاستبقاء، والتوافر، مما يؤدي إلى انفصال الطلاب أو عدم الكفاءة. يعتمد النظام المقترح على تقنيات تعلم الآلة — شبكات الذاكرة طويلة قصيرة المدى (LSTM) وتعلم التعزيز — (RL) لتعديل خطط الحفظ والمراجعة ديناميكيًا بناءً على أداء الطالب الفعلي والبيانات التاريخية. تُحلّل نماذج LSTM أنماط التعلم المتسلسلة للتنبؤ بأهداف الحفظ الأساسية، بينما يُعدِّل RL هذه الأهداف استجابةً لتعليقات التلاوة، مُحسِّنًا التوازن بين حفظ الأجزاء الجديدة والمراجعة .تكمن المنهجية في دمج بنية قواعد بيانات قوية (PostgreSQL)لإدارة الملفات الشخصية للطلاب، والمحتوى القرآني (السور والآيات)، وجلسات التلاوة، مما يضمن تدفقًا سلسًا للبيانات بين نماذج الذكاء الاصطناعي وواجهات المستخدم. سمح التعاون مع شركة تعليم قرآنية بالوصول إلى بيانات تاريخية مُنظمة، مكَّنت من تدريب نماذج LSTMعلى أنماط حفظ واقعية. كشفت الاختبارات دقة تنبؤية متوسطة، حيث فسرت LSTM 47%، و70%، و64% من التباين في مهام الحفظ الجديد، والمراجعة البسيطة، والمراجعة الشاملة على التوالي. لكن ارتفاع معدلات الأخطاء في المراجعات الشاملة (1,265 حرفًا) يُظهر فرصًا للتحسين. بُنِيَ النظام الخلفي (Backend) باستخدام Python و TensorFlowو SQLAlchemy لدعم التعديلات الفورية، بينما تقدم الواجهة الأمامية (Bootstrap) أدوات سهلة للمعلمين لتتبع التقدم وتحديث الخطط .تشمل النتائج الرئيسية نظامًا مرنًا يتكيف مع المسارات التعليمية الفردية، ويعزز الاستبقاء عبر التكرار المتباعد، ويوفر رؤى قابلة للتنفيذ للمعلمين. تركز التطويرات المستقبلية على نشر النظام كتطبيق ويب/جوال قابل للتطوير، ودمج التعرف الآلي على الصوت لتقييم التلاوة، وتوسيع أدوار المستخدمين لتشمل المشرفين وأولياء الأمور. يُظهر هذا المشروع إمكانات تحويلية للتقنية في التعليم القرآني، عبر إطار عمل مرن يعتمد على البيانات لتمكين المتعلمين عالميًا، مُجَسِّدًا الجسر بين إمكانات الذكاء الاصطناعي والاحتياجات التربوية.

Contents

List Of Figures

List Of Tables

CHAPTER 1

INTRODUCTION

1.1. Introduction

In many traditional Quran learning environments, students are expected to follow a fixed memorization schedule, which doesn't account for their unique abilities. As a result, some students may struggle to keep up, while others aren't being challenged enough. This project explores how AI can offer a more personalized approach to Quran memorization, improving the learning experience for each student.

1.2. Aim

This project aims to propose and design a machine-learning model that provides personalized and adaptive Quran memorization plans for individual students. Using artificial intelligence, the system will analyze user-specific data, such as current memorization level and student performance, to create an efficient, manageable, and flexible memorization plan. The goal is to enhance the students' progress through optimized scheduling, making memorization more structured, personalized, and achievable.

1.3.Problem Definition

Memorizing the Quran is a cherished tradition in Islam, with countless Muslims dedicating themselves to this sacred pursuit. However, in today's world, memorizing the Quran presents significant challenges for students. Most Quran memorization programs follow fixed schedules that do not account for individual differences in learning abilities. As a result, students with varying memorization speeds, cognitive abilities, and learning

preferences are forced to follow a standardized approach that may not suit their unique needs.

In this traditional approach, students are typically expected to memorize a set portion of the Quran each day and review previously memorized sections periodically. While this approach may work for some, many students face difficulties keeping up with the assigned pace. Some struggle due to personal factors such as limited time availability, weaker memorization ability, or slower memorization speed. These students may fall behind, becoming discouraged and less motivated, which can ultimately affect their progress and confidence. On the other hand, students who can memorize quickly and efficiently often find themselves under-challenged, leading to disengagement and limited learning. Both scenarios highlight the inefficiencies in the fixed memorization plans.

Furthermore, one of the key challenges in Quran memorization is retention. Even if students manage to memorize new material, retaining it over time requires regular revision. Traditional systems tend to implement fixed revision schedules that do not consider individual retention capabilities. Without an adaptive mechanism to adjust revision plans based on each student's needs, students may either forget previously memorized sections due to inadequate review or spend too much time revising verses they have already mastered, limiting their overall progress.

The absence of intelligent feedback systems makes the lack of personalization in Quran memorization programs worse. Teachers often rely on their judgment to assess a student's progress, which can be subjective and inconsistent. There's no structured way to adjust memorization plans based on a student's ongoing performance, leaving many students frustrated and stuck in ineffective learning.

Given these challenges, there is a growing need for an intelligent system to offer a more personalized approach to Quran memorization. Artificial intelligence and machine learning technologies have the potential to transform this process by analyzing individual learning patterns and generating tailored memorization plans. These technologies can assess a student's current level, pace, and performance, adjusting the memorization and revision schedules dynamically to ensure that each student receives the appropriate level

of challenge and support. Such a system would address the issue of varying memorization speeds and enhance long-term retention through customized revision strategies.

1.4. Target Users

1. **Students of Quran Memorization**: Individuals of all ages and backgrounds aiming to memorize the Quran, whether beginners, intermediates, or advanced learners. These students may vary in memorization speed, learning ability, and daily availability, making personalized plans critical for effective progress.

2. **Quran Teachers/Instructors**: Educators who supervise students' Quran memorization can use the system to track student progress, get performance insights, and adjust learning strategies for each student more effectively.

3. **Islamic Educational Institutions**: Islamic schools and online Quran learning platforms that want to offer their students a modern and personalized Quran memorization tool, improving overall learning outcomes.

4. **Parents/Guardians**: Those responsible for guiding their children through Quran memorization, seeking a tool that provides structured, personalized plans to ensure effective learning and retention.

1.5. Suggested Solution

The proposed solution is an **AI-based Quran Memorization Plan Generator** that designs personalized memorization and revision plans based on a student's unique learning profile, performance data, and retention ability. The system will analyze user-specific inputs such as current memorization level, daily availability, and revision needs,

to generate and adapt a flexible memorization plan that balances new verses with structured revision.

1.5.1. Key Features of the Suggested Solution:

1. **Personalized Memorization Plans:** Create customized plans for each student, considering their pace, availability, and progress. The system generates new verses to memorize, alongside minor and major revisions based on what the student has already memorized.
2. **Performance-Based Adjustments:** The AI model adjusts memorization, and revision plans automatically based on the student's performance, including speed, accuracy, and retention rates. The system can scale up or slow down the workload to match the students' capabilities.
3. **Adaptive Revision Plans:** The system ensures that students regularly review previously memorized verses, using personalized intervals for revision. Minor revisions cover recently memorized verses, while major revisions periodically improve the retention of large parts of previously memorized sections.
4. **Flexible and Dynamic Scheduling:** The system adapts to changes in the student's availability or challenges, allowing them to pause, adjust, or modify their memorization plan without losing progress.

1.6.Initial System Overview

The project focuses on developing an AI model that generates personalized Quran memorization plans based on a student's performance data. The model will operate without a complex system or UI, but with a simple interface to input performance data and output the next memorization plan. Below is a detailed overview of the project:

1.6.1. Objective

The AI model will create three types of plans:

- **New Memorization Plan**: Assign new Quranic verses for students to memorize.

- **Minor Revision Plan**: Focuses on revising recently memorized portions.
- **Major Revision Plan**: Periodic review of larger portions of memorized Quran to ensure long-term retention.

1.6.2. Key Functionalities

The AI model will have the following functionalities:

1. **Initial Training on Historical Data**: Using a dataset that contains records of student recitations, the model will learn patterns from past performances.
2. **Real-Time Plan Adaptation**: After each student's recitation, the model will analyze performance data (e.g., mistakes made, and whether the recitation was accepted as valid) to adjust future memorization and revision plans.
3. **Dynamic Plan Generation**: The model continuously refines its predictions, generating new, minor, and major revision plans based on ongoing performance tracking.

1.6.3. Input Data

- **Training Dataset**: Contains recitation records of students, with details such as:
  - Portion requested for recitation.
  - Number of mistakes made.
  - Whether the recitation was accepted as valid.
- **Real-Time Performance Data**: Captured after each recitation session, this data will be used to modify the student's plans. It includes:
  - Performance on the most recent recitation.
  - Mistake count.
  - Feedback on the validity of the recitation.

1.6.4. Output

- **Memorization Plan**: Assign new Quranic verses based on the student's capacity.
- **Minor Revision Plan**: Suggests portions of recently memorized content for revision to reinforce short-term memory.
- **Major Revision Plan**: Periodically suggests larger sections for review to support long-term retention.

1.6.5. Development Activities

1.6.5.1. Requirements Specification

- Identify key metrics for evaluating student performance, such as number of mistakes, pace of recitation, and memorization capacity.
- Define how often plans should be adjusted based on recitation feedback.

1.6.5.2. Data Collection & Preprocessing

- Data Collection: Collect historical data on student recitations, including mistakes and feedback.
- Data Preprocessing: Clean and structure the dataset to ensure consistent input for model training, including handling missing data, normalizing performance metrics, and organizing data by individual student records.

1.6.5.3. AI Model Design and Training

- Model Selection: Choose an appropriate machine learning model (e.g., decision tree, neural network) capable of handling sequential data for learning progression and retention patterns.
- Training: Use the preprocessed dataset to train the AI model. The model will learn from past performance data, identifying trends in student memorization speed, error rates, and retention.

1.6.5.4. Model Testing and Validation

- Testing: Validate the AI model using unseen data to ensure it provides accurate memorization plans and adjusts based on student performance.
- Evaluation Metrics: Use evaluation metrics such as accuracy in predicting next verses and F1-score for mistake prediction to ensure robustness.

1.6.5.5. Real-Time Plan Adjustment Implementation

- Integration: Incorporate a feedback loop where new student performance data is fed into the model after each recitation, allowing real-time updates to memorization and revision plans.
- Performance Monitoring: Continuously monitor the model's predictions to ensure the plans are effectively adapted to each student’s learning pace.

1.6.5.6. Continuous Learning & Improvement

- Use reinforcement learning or a similar approach to enable the AI model to learn and improve its predictions over time based on feedback from real-time recitations and evaluations.

1.7. Project Scheduling

Waterfall Phase	Milestones	Tasks	Week
Analysis	Initial Project Proposal	Finalize project title, objectives, and scope	Week 3
		Prepare project significance overview	
		Conduct preliminary research	Week 4
		Gather sources for the literature review	
		Prepare slides for initial presentation	Week 5
		Draft and refine the project proposal	
	Project Proposal and Literature Review	Begin writing Report #1	Week 6
		Research existing Quran memorization tools and AI applications	
		Conduct literature review	Week 7
	Initial Requirements Analysis	Conduct initial requirements analysis	Week 8
		Identify user needs and technical requirements	
		Finishing writing Report #1	
	Presentation #1 Preparation	Prepare slides for Presentation #1	Week 9
Design	Detailed Requirements and System Design	Conduct detailed requirements analysis	Week 10
		Start data collection and description	
		Begin writing Report #2	Week 11
		Begin system design and architecture	
		Develop initial prototypes	

	Report#2 and Presentation#2 Submission	Finishing writing Report #2	Week 12
		Prepare slides for Presentation #2	
	Final Report Submission	Compile and finalize all chapters for the final report	Week 13
		Ensure system design and prototyping documentation is complete	
		Review and refine the final report for submission	
		Complete all necessary revisions	
		Final Report (All Chapters) Submission	Week 14
	Poster Design and Preparation	Design the project poster	
		Finalize the key points and visuals for the poster presentation	Week 15
	Final Presentation Preparation	Prepare the slides for the final presentation	
		Review all phases of the project to present key findings, challenges, and outcomes	

Table 1: Project Scheduling

CHAPTER 2

LITERATURE REVIEW

2.1.Introduction

Integrating artificial intelligence (AI) into personalized learning environments has become a transformative approach in modern education. AI systems provide dynamic, real-time adaptation of learning plans to individual needs, making learning more efficient and targeted. In Quran memorization, an area that requires continuous revision and personalized pacing, traditional methods often fall short in addressing the unique requirements of learners. This literature review explores themes around the gaps in existing Quran memorization technologies, personalized learning, and adaptive AI-driven educational models, positioning them within the scope of developing an AI-based Quran Memorization Plan Generator.

2.2.Gaps in Quran Memorization Technologies

Despite the increasing use of technology in Quran memorization, significant gaps remain, particularly in personalized, adaptive systems. Most current tools offer static memorization schedules or simple tracking features, lacking the ability to adjust to the individual needs of each learner.

Haryono et al. (2023)[1] provide a comprehensive review of Quran memorization technologies and highlight the absence of AI-driven tools that offer real-time adaptation and personalization. Existing systems are often limited to tracking progress and providing access to Quranic text but fail to dynamically adjust to the learner's memorization pace or offer tailored revision schedules. This gap points to the need for more intelligent systems to analyze student data and adjust the learning path accordingly.

The proposed AI-based Quran Memorization Plan Generator would fill this gap by offering a more personalized and adaptive approach. Using AI to create dynamic, flexible

plans based on individual performance would make Quran memorization more accessible and effective, especially for students with varying memorization and retention abilities. By addressing the limitations identified by **Haryono et al. (2023)**[1], the project aims to revolutionize how students approach Quran memorization, making the process more structured and personalized.

2.3. Personalized Learning: The Role of AI in Education

Traditional teaching methods often follow strict, standardized approaches that may not consider individual student differences. However recent advancements in AI offer solutions that can dynamically adjust to each learner's pace, abilities, and progress.

Maghsudi et al. (2021)[2] discuss how AI can provide personalized learning experiences by analyzing student data to deliver tailored content and feedback. This aligns with the needs of Quran memorization, where learners progress at different rates, and fixed memorization schedules can hinder learning. In Quranic studies, an AI-driven system can continuously monitor a student’s progress, adjusting both the number of new verses assigned and the frequency of revision sessions to optimize learning outcomes. Similarly, **Anil et al. (2019)**[3] proposes dynamic learning plan generators that adjust to individual learning profiles, emphasizing how real-time student data can influence personalized learning paths. This dynamic approach, when applied to Quran memorization, could enable learners to move at their own pace while ensuring consistent review of previously memorized content.

The significance of personalized learning systems in Quran memorization is that they break away from one-size-fits-all approaches, offering a more flexible, student-centered model. This would allow each student to follow a plan that matches their memorization ability, ensuring both progression and retention, key challenges in Quran learning.

2.4. Adaptive Learning Systems and Real-Time Feedback

The concept of adaptive learning systems builds on personalization by adding continuous, real-time adaptation based on performance. In Quran memorization, this is particularly important because learners often need to balance learning new material with revisiting previously memorized content to strengthen retention. AI's capability to dynamically adjust learning paths in real time is a key enabler of more effective Quran memorization practices.

Somasundaram et al. (2020)[4] introduce an AI-based system for managing learning paths, where student performance is tracked, and plans are adjusted according to real-time feedback. This continuous adaptation is especially relevant to Quran memorization, where an AI system could monitor how well students are retaining verses and suggest revision sessions when needed.

Oussama et al. (2022)[5] also emphasize the importance of adaptability in AI-driven education systems, suggesting that personalized learning paths should evolve based on ongoing assessments of student progress.

For Quran memorization, a similar model could be developed where AI tracks factors such as recitation accuracy, speed of memorization, and long-term retention. This would ensure that memorization plans remain responsive to the learner's needs, providing a flexible and sustainable learning journey. By automatically adjusting revision intervals and assigning new verses when appropriate, AI systems can help prevent learners from becoming overwhelmed, thus improving memorization effectiveness.

2.5. The Importance of Revision in Quran Memorization

A significant challenge in Quran memorization is retention, and effective revision strategies are essential for long-term success. Research into AI-based learning systems shows that spaced repetition and revision are critical components of adaptive learning models, ensuring that learners retain what they have memorized.

Zhang et al. (2020)[6] focus on the implementation of personalized learning systems and highlight the importance of structured revision to reinforce learning. In the Quran memorization, the need for a consistent review of previously learned material cannot be overstated. An AI-based system can be designed to incorporate these findings by scheduling both minor and major revisions based on student performance data. The system would ensure that learners are regularly revisiting past lessons to reinforce their memorization, preventing long-term forgetting. By combining new memorization with periodic review, the AI-based system could effectively support both short-term learning goals and long-term retention.

This combination of new learning and targeted revision allows students to maintain a steady pace of progress while ensuring that older content is regularly revisited. It addresses a key issue in Quran memorization—balancing the need for continuous progress with the requirement for periodic review to reinforce long-term memory.

2.6.Conclusion

The literature on AI in education consistently highlights the potential of personalized, adaptive learning systems to enhance student outcomes by tailoring the learning experience to individual needs. In the context of Quran memorization, where learners require both flexibility and structure, the development of an AI-based memorization plan generator addresses several key challenges. The studies reviewed provide a strong theoretical foundation for applying AI to personalize learning, optimize revision schedules, and improve retention. By incorporating real-time performance feedback and adaptive scheduling, this project can significantly improve the efficiency and effectiveness of Quran memorization, bridging the gap identified in current Quran memorization technologies.

CHAPTER 3

DATA COLLECTION AND ANALYSIS TECHNIQUES

3.1.Methods to Find Information

To gather the comprehensive information required for our project, we employed a multifaceted approach that combined research with insightful interviews with subject matter experts from a Quranic education management company and teachers of the Quran.

3.1.1. Research

To collect system requirements, we utilized research as a foundational method for identifying functional, non-functional, hardware, and software needs. Research allowed us to define functional requirements by studying the key features needed for personalized Quranic memorization, such as tracking student progress, creating adaptive memorization plans, and supporting specific learning methodologies. It also provided insights into non-functional requirements, helping us set clear standards for system performance, scalability, usability, and data security.

Additionally, research guided us in understanding the hardware requirements, such as the computational resources necessary to train and deploy our models effectively, as well as the storage needed for handling large datasets. On the software side, it helped us determine the tools, frameworks, and platforms best suited for developing and deploying the system efficiently. By systematically analyzing available information, research enabled us to establish a comprehensive and realistic set of system requirements aligned with the project's goals.

3.1.2. Interviews with subject matter experts

Interviews with subject matter experts played a crucial role in gathering comprehensive system requirements, including functional and non-functional aspects.

These discussions provided practical, experience-driven insights that were instrumental in defining the system's functional requirements. Experts highlighted key features necessary for effective Quranic memorization, such as personalized learning plans, real-time feedback on recitations, and tools for tracking student progress. Their input also clarified the importance of incorporating features that address common challenges, like frequent errors in pronunciation or retention, ensuring the system aligns closely with real-world teaching practices.

For non-functional requirements, experts emphasized the need for a user-friendly interface suitable for both teachers and students, as well as the importance of system reliability and accuracy in evaluating recitations. They also provided feedback on the system's response times, scalability for handling multiple users and maintaining data privacy, especially for student performance records.

3.2.Requirement Specification

3.2.1. Functional Requirements

The functional requirements of the model are designed to support adaptive and personalized learning experiences for each student. The main functional requirements include:

1. **Generate Daily Memorization Plans**:
    - The model should generate a personalized daily plan for memorization and revision for each student based on his historical performance and current needs.
2. **Adjust Plans in Real-Time**:
    - The system should dynamically adjust the memorization and revision goals based on the student's last recitation session, including errors and accuracy. Real-time adaptation is achieved through Reinforcement Learning (RL).

3. **Track and Record Performance Data**:
    - The system should record the data of each recitation session, including the verses recited, errors, and the type of recitation (new memorization, major revision, minor revision).
4. **Historical and Cluster-Based Analysis**:
    - For new students for which there is no personal historical data, the model should use performance data from similar students to determine a suitable starting goal.
5. **Provide Performance Feedback**:
    - Provide feedback to teachers based on errors and accuracy and guide them to areas that need to be revised or improved.

3.2.2. Non-Functional Requirements

These requirements ensure the usability, performance, and reliability of the system :

1. **Usability**:
    - The model should have a simple user interface which will make it easier for teachers to enter student data and retrieve daily plans.
2. **Performance**:
    - Adjustments should occur in real-time within seconds to ensure a smooth and responsive operation process.
3. **Scalability**:
    - The model should be able to handle several students at once, which will allow the system to grow as the number of users increases.
4. **Reliability**:
    - The system should work constantly while ensuring data integrity and accuracy in generating plans and tracking performance.

5. **Maintainability**:
    - The architecture of the system should be modular and easy to update, which allows it to adjust and improve efficiently.

3.2.3. Data Requirements

The dataset needs to capture detailed records of each student's recitation history and real-time performance data:

1. **Student Data**:
    - Basic information like student ID, name, and memorization start date.
    - Historical recitation data, including verses recited and type of recitation.
2. **Session Performance Data**:
    - Data on recitation sessions, such as date, type, errors, acceptance, and pace (letters and pages recited).
3. **Historical and Cluster-Based Data**:
    - Aggregated performance data from similar students, organized by clusters, for new sections with no specific student history.
4. **Generated Feedback Data**:
    - Referring to the data that captures any feedback or adjustments made to a student's daily memorization and revision plan based on their recent recitation performance. This feedback includes information on how the system responded to the student's session—whether the target was adjusted, the type of adjustment made, and the reasons behind it.

3.2.4. Software Requirements

The project requires a combination of machine learning libraries, data management tools, and programming environments to develop, deploy, and maintain the system.

1. **Programming Languages**:
    - **Python**: Primary language for model development, data processing, and integration.
2. **Machine Learning Libraries**:
    - **TensorFlow or PyTorch**: For building the machine learning model.
    - **scikit-learn**: For clustering similar students based on performance metrics.
3. **Database Management**:
    - **PostgreSQL**: To store and retrieve student records, session logs, and performance metrics.
4. **Data Processing Libraries**:
    - **Pandas, NumPy**: For data cleaning, manipulation, and analysis.
5. **Environmental and Deployment Tools**:
    - **Jupyter Notebook**: For model development and testing.
    - **Docker**: For containerizing the application for easy deployment and scalability.

3.2.5. Hardware Requirements

1. **Server Requirements**:
    - **CPU**: Multi-core processors to handle model training and data processing.
    - **GPU**: Required for efficient training of the models as we are working with a large dataset.
    - **RAM**: Minimum 16GB to support data handling, model training, and database operations.

2. **Local System Requirements**:
    - Personal computers for development with at least 8GB RAM and a modern CPU.
3. **Cloud Infrastructure (Optional)**:
    - **AWS** or **Google Cloud Platform**: For scalable deployment and storage of large datasets, with GPU capabilities for model training.

3.2.6. Security Requirements

Security is essential to protect sensitive student data and ensure the system's integrity. Key security requirements include:

1. **Access Control**:
    - Role-based access control (RBAC) to limit data access based on user roles, allowing only authorized teachers and administrators to access or update student records.
2. **Data Backup and Recovery**:
    - Regular backup of the database to prevent data loss, with a recovery plan in case of accidental data deletion or corruption.
3. **Authentication**:
    - Implement secure login and authentication methods for users to access the system, such as two-factor authentication (2FA) for added protection.

3.3. Acquiring Data

To collect the data, we needed for our project, we teamed up with a company that specializes in managing Quranic education. This company's system keeps detailed records of student recitations, including their performance, memorization progress, and revision habits. By working with them, we got access to this organized data, which became the backbone of our personalized Quran memorization model.

We started our data-gathering journey by chatting with the company's team. These conversations gave us valuable insights into how their system tracks and evaluates student performance, manages recitation logs, and organizes memorization and revision cycles. The information they shared helped us understand the real-world aspects of Quranic education, which was crucial for identifying relevant data points and grasping the context of recitation records. This knowledge shaped our approach, ensuring that our data collection process aligned with the specific needs of our model.

While we also searched online for publicly available data on student recitation patterns and memorization processes, we found limited resources that fit our project's unique requirements. Most online datasets either lacked the necessary detail or weren't specifically focused on Quranic memorization. As a result, our collaboration with the Quranic education company became essential, providing us with both the required data and an understanding of its practical application in educational settings. This partnership not only enriched our dataset but also guided our methodology, ensuring that our project remained closely connected to real-world Quranic memorization practices.

3.4.Dataset Overview

The dataset contains three main tables that provide a complete view of each student's journey in memorizing and revising the Quran. These tables are:

3.4.1. Student Lesson History

It tracks individual recitation sessions for each student and explains the specific verses recited, the type of recitation, and performance indicators. The main columns include:

- **student_id**: Defines each student uniquely.
- **start_verse_id** and **end_verse_id**: Determines the range of verses recited in each session.
- **pillar_id**: Indicates the type of recitation (for example, 1 for new memorization, 2 for major revision, 3 for minor revision). This column is

useful for classifying types of recitation, which helps the model distinguish between new memorization sessions and revision.

- **teacher_id**: Defines each teacher uniquely.
- **date_of**: Date of the recitation session.
- **letters_count** and **pages_count**: Describes the amount of recitation based on the range of verses recited in each session. These columns are important for tracking the amount of conservation.

B	C	D	E	F	G	H	I
student_id	start_verse_id	end_verse_id	pillar_id	teacher_id	date_of	calculated_letters_count	calculated_pages_count
424	6231	6225	3	422	8/20/2023	257	1.000001
452	6080	5993	3	449	8/21/2023	2154	4.590018
479	6231	5932	3	449	8/22/2023	5580	13.455782
334	3971	3793	2	523	8/23/2023	3187	5.413072
331	5623	5640	2	522	8/24/2023	263	0.516698
331	5623	5640	2	83	8/24/2023	263	0.516698
331	6231	6234	2	523	8/24/2023	66	0.256809
332	6231	6226	2	522	8/24/2023	132	0.513619
332	5673	5702	3	522	8/24/2023	519	1.000003

Figure 1: Student Lesson History Table

3.4.2. Surahs

It stores constant information about each Surah (chapter), including the number of verses. The main columns include:

- **surah_id**: A unique identifier for each Surah.
- **name**: The name of the Surah in the Quran.
- **no_verses**: The total number of verses in each Surah.

A	B	C
surah_id	name	no_verses
1	الفاتحة	7
2	البقره	286
3	آل عمران	200
4	النساء	176
5	المائده	120
6	الأنعام	165
7	الأعراف	206
8	الأنفال	75
9	التوبة	129
10	يونس	109
11	هود	123
12	يوسف	111
13	الرعد	43
14	إبراهيم	52
15	الحجر	99
16	النحل	128
17	الإسراء	111
18	الكهف	110
19	مريم	98
20	طه	135
21	الأنبياء	112
22	الحج	78

Figure 2: Surah Table

3.4.3. Verses

It provides verse-specific details, including the order in the Quran and each Surah, as well as the number of letters and page locations. The main columns include:

- **verse_id**: A unique identifier for each verse.
- **surah_id**: Connects each verse with its own Surah.
- **order_in_quraan**: The sequential order of the verse in the Quran.

- **reverse_index**: The reverse order of verses.
- **order_in_surah**: The sequential order of the verse in the Surah.
- **page_no**: The location of the page where the verse is located.
- **letters_count**: The total number of letters in each verse, used to analyze the speed of memorization.
- page_no, letters_count, and weight_on_page: Details on where the verse is located, its length, and its weight on the page.

A	B	C	D	E	F	G	H	I
verse_id	surah_id	begin_verse	order_in_quraan	reverse_index	order_in_surah	page_no	letters_count	weight_on_page
1	1	بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ	1	6230	1	1	19	0.132867
2	1	ٱلْحَمْدُ لِلَّهِ رَبِّ	2	6231	2	1	18	0.125874
3	1	ٱلرَّحْمَٰنِ ٱلرَّحِيمِ	3	6232	3	1	12	0.083916
4	1	مَٰلِكِ يَوْمِ ٱلدِّينِ	4	6233	4	1	12	0.083916
5	1	إِيَّاكَ نَعْبُدُ وَإِيَّاكَ	5	6234	5	1	19	0.132867
6	1	ٱهْدِنَا ٱلصِّرَٰطَ ٱلْمُسْتَقِيمَ	6	6235	6	1	19	0.132867
7	1	صِرَٰطَ ٱلَّذِينَ أَنْعَمْتَ	7	6236	7	1	44	0.307692
8	2	بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ	8	5944	1	2	22	0.121547
9	2	ذَٰلِكَ ٱلْكِتَٰبُ لَا	9	5945	2	2	27	0.149171
10	2	ٱلَّذِينَ يُؤْمِنُونَ بِٱلْغَيْبِ	10	5946	3	2	47	0.259669
11	2	وَٱلَّذِينَ يُؤْمِنُونَ بِمَآ	11	5947	4	2	52	0.287293
12	2	أُوْلَٰٓئِكَ عَلَىٰ هُدًى	12	5948	5	2	33	0.18232
13	2	إِنَّ ٱلَّذِينَ كَفَرُوا۟	13	5949	6	3	47	0.083929
14	2	خَتَمَ ٱللَّهُ عَلَىٰ	14	5950	7	3	53	0.094643
15	2	وَمِنَ ٱلنَّاسِ مَن	15	5951	8	3	47	0.083929
16	2	يُخَٰدِعُونَ ٱللَّهَ وَٱلَّذِينَ	16	5952	9	3	49	0.0875
17	2	فِى قُلُوبِهِم مَّرَضٌ	17	5953	10	3	51	0.091071
18	2	وَإِذَا قِيلَ لَهُمْ	18	5954	11	3	43	0.076786
19	2	أَلَآ إِنَّهُمْ هُمُ	19	5955	12	3	29	0.051786
20	2	وَإِذَا قِيلَ لَهُمْ	20	5956	13	3	77	0.1375
21	2	وَإِذَا لَقُوا۟ ٱلَّذِينَ	21	5957	14	3	73	0.130357
22	2	ٱللَّهُ يَسْتَهْزِئُ بِهِمْ	22	5958	15	3	34	0.060714
23	2	أُوْلَٰٓئِكَ ٱلَّذِينَ ٱشْتَرَوُا۟	23	5959	16	3	57	0.101786
24	2	مَثَلُهُمْ كَمَثَلِ ٱلَّذِى	24	5960	17	4	72	0.125217
25	2	صُمٌّۢ بُكْمٌ عُمْىٌ	25	5961	18	4	19	0.033043
26	2	أَوْ كَصَيِّبٍ مِّنَ	26	5962	19	4	86	0.149565
27	2	يَكَادُ ٱلْبَرْقُ يَخْطَفُ	27	5963	20	4	102	0.177391
28	2	يَٰٓأَيُّهَا ٱلنَّاسُ ٱعْبُدُوا۟	28	5964	21	4	53	0.092174
29	2	ٱلَّذِى جَعَلَ لَكُمُ	29	5965	22	4	99	0.172174
30	2	وَإِن كُنتُمْ فِى	30	5966	23	4	78	0.135652

Figure 3: Verses Table

3.5.Dataset Utilization in the Project

The detailed structure of the dataset allows it to support a personalized Quran memorization plan by providing:

- **Deriving Recitation Patterns for Model Training**:
    - The dataset provides training examples of common recitation patterns, average memorization speeds, and frequent errors.

- This data allows the model to generalize typical learning behaviors, making it more accurate in guiding both new and experienced students through personalized recitation plans.

- **Collective Cluster-Based Data**
  - In addition to individual data, the dataset enables the calculation of average recitation pace and performance across groups of students at similar levels.
  - These averages provide a foundation for creating initial plans for new students or those without a personal recitation history in certain surahs. By using typical pace data from students with similar progress patterns, the model can set realistic starting goals even before personalized patterns are established.

This refined structure clarifies that initial plans for students without history rely on collective averages from similar students, while historical data and recitation patterns aid in model training and personalization.

CHAPTER 4

ARCHITECTURE AND DESIGN

4.1. Machine Learning Approach

Memorizing the Quran is a process that requires sequential understanding and the ability to adapt to the progress of each student. To meet these needs, we recommend a hybrid approach that combines Long Short-Term Memory (LSTM) networks and Reinforcement Learning (RL). This combination is ideal because LSTM is excellent at recognizing patterns in sequential data, such as the student's historical memorization performance, while RL allows the model to adapt dynamically to real-time progress. Together, these methods provide a balanced approach that supports continuous growth in memorization and adapts to each student’s unique pace and retention needs.

In addition, an algorithm will be used to evaluate the overall performance of students in various recitation parts, providing basic metrics. These metrics will help prepare initial plans for students who do not have previous recitation data, allowing a personal starting point based on average performance. This approach ensures a steady growth in memorization while adapting to the learning path of each student.

4.2. What are LSTM and RL?

4.2.1. Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) designed to handle sequential data and retain dependence on previous information for long periods[7]. This is very useful for tasks that require knowledge of past information to understand or predict future results. The LSTM network relies on memory cells to capture patterns over time, making

it suitable for analyzing students' historical memorization data. By identifying pace, retention, and accuracy trends, LSTM helps the model set personalized, data-driven goals that align with the student's learning style and past performance.

4.2.2. Reinforcement Learning (RL)

Reinforcement learning is a type of machine learning in which an agent learns to make decisions by interacting with the environment[8]. The agent receives feedback in the form of rewards or penalties based on its actions, directing it towards achieving a specific goal over time. Through trial and error, the agent learns how to optimize cumulative rewards, which allows it to adapt its behavior to achieve better results. In the context of Quran memorization, RL can adjust daily goals in real-time based on each student's performance, promoting a flexible and responsive learning experience.

4.3.Why use LSTM and RL in our project?

4.3.1. Long Short-Term Memory (LSTM)

The reasons to use Long Short-Term Memory (LSTM) are:

- **Capturing Sequential Dependencies**: Quran memorization is a cumulative process where each recitation session builds on previous sessions. LSTM models can capture these dependencies by processing sequences of past performance data, such as the number of verses recited and pace. This enables the model to learn patterns over time, allowing it to generate a basis for each student's daily memorization and revision goal.

- **Predicting Memorization Trajectories**: By analyzing historical data for each student, the LSTM model can recognize patterns in learning speed, consistency, and areas that require more effort[7]. For example, if a student shows steady progress in memorizing about five verses per session with few mistakes, LSTM can predict that the student may be ready for a small increase

in the daily goal. Conversely, if previous data shows repeated challenges in certain surahs or verses, the model can adjust the plan to ensure that those areas are reviewed more thoroughly.

- **Foundation for Baseline Plans**: The LSTM generates a personalized baseline plan for each student, setting daily goals for new memorization and revision based on past patterns. This plan is initial and adaptable, as it is aligned with the unique speed of each student and his retention needs. Moreover, the RL component optimizes the plan so that the model can keep up with the performance in real-time.

4.3.2. Reinforcement Learning (RL)

The reasons to use Reinforcement Learning (RL) are:

- **Real-Time Plan Adjustments**: Unlike LSTM, which generates a basic plan based on historical data, RL enables real-time adjustments. After each recitation session, the RL model reviews performance metrics, such as the number of errors, missed sessions, or student accuracy, and updates the plan accordingly. For example, if a student completes a session with high accuracy and few errors, the RL model may slightly increase the goal for new verses for the next session. Conversely, if accuracy decreases or errors increase, the agent may prioritize additional review sessions.
- **Adaptability to Immediate Performance**: RL is characterized by responding to immediate changes in performance. For example, if a student has not reviewed a particular Surah for a long time, the RL agent can add that Surah to the Daily plan, even if it is not included in the basic plan created by LSTM. This ensures that the student does not lose retention of previously memorized parts while working on new material.
- **Reinforcement Through Rewards and Penalties**: Reinforcement through rewards and punishments: the RL model uses a reward system to guide

adjustments in the daily plan. Positive performance metrics, such as high accuracy and consistency, may lead to rewards such as adding new verses for memorization. Conversely, penalties, such as repeating mistakes, may lead to a reduction in the daily target or the introduction of additional review sessions. This system based on rewards and penalties helps to balance progress and retention, enabling the student to progress without feeling overwhelmed.

4.4. Key Model Inputs

- **LSTM Inputs**: LSTM in this project uses existing columns from the historical dataset, such as letters_count, pages_count, and pillar_id, which represent the amount and type of verses recited in previous sessions.
- **RL Agent Inputs**: The RL agent uses the data generated newly in each recitation session, which is stored in a separate database with additional columns added specifically for tracking performance metrics in real-time.

4.5. Data Processing Workflow

1. **Preprocessing**:
    - Grouping daily recitation data into sequences for each student, enabling the LSTM to learn from past performance patterns.
2. **Model Training**:
    - The LSTM is trained on these sequences to estimate an initial daily memorization and revision plan based on historical trends.
    - For new students with no history, an algorithm calculates average performance metrics from other students to generate initial plans based on common recitation patterns.
3. **RL Agent for Real-Time Adjustment**:

- The RL agent dynamically adjusts the plan:
  - **New Surahs with No History**: Combines general pace with cluster-based averages to set a suitable starting target.
  - **Previously Memorized Surahs**: Integrates the student's general and surah-specific pace for revisiting familiar sections.
- The RL agent improves retention by adjusting the quantities according to recent errors, missed sessions, and time since the last review.

4.6. Summary of the Approach

The combined approach of LSTM and RL, supported by an algorithm that uses average performance metrics, creates a flexible and personalized system for memorizing the Quran. This approach adapts to the historical performance of each student, real-time progress, and data from similar students. By customizing the learning plan with targeted adjustments, the model aligns with the pace of each student, focusing on difficult areas and enhancing memory retention. This adaptive approach enables confident progress and provides a balanced and responsive memorization experience designed to meet changing needs.

4.7. Needed Skills

4.7.1. Main Skills Needed

4.7.1.1. Machine Learning and Deep Learning

The backbone of this project relies on machine learning—specifically, using Long Short-Term Memory (LSTM) networks and Reinforcement Learning (RL). LSTM networks are vital for understanding patterns in the sequence of a student's past recitations, while RL enables the system to adapt to each student's real-time performance, adjusting plans to meet their current needs. LSTM helps us understand sequential data, which is perfect for tracking memorization over time. On the other hand,

RL allows the system to respond to a student's real-time progress by adjusting daily memorization plans. Together, these methods provide a personalized, data-driven learning experience that can flexibly adapt to each student's needs.

4.7.1.2. Data Processing and Analysis

High-quality data is at the heart of this project, as our model will rely on historical recitation records and real-time performance metrics to make informed recommendations. Data processing skills are needed to clean, organize, and transform this data into suitable for our models. The input data must be clean, well-organized, and contain relevant features for the model to make accurate predictions and adjustments. This ensures the LSTM and RL models can make meaningful decisions based on each student's recitation patterns, pacing, and accuracy levels. Without strong data processing, the model's output may be unreliable or difficult to interpret, hindering the project's success.

4.7.1.3. Database Management and SQL

This project requires effective storage and retrieval of both historical and real-time performance data. Database management skills are needed to create and organize tables for storing recitation history, performance metrics, and dynamically generated recommendations. The project combines existing historical data with continuously updated, session-based performance metrics. Database skills are crucial to efficiently organize, access, and manage this information, ensuring that the system can easily retrieve and store data for real-time adjustments. A well-designed database structure allows the model to access necessary information instantly, facilitating smooth operations.

4.7.1.4. Python Programming

Python is the primary programming language for machine learning and data processing, making it essential for implementing and integrating LSTM and RL models. Python provides the libraries we need (like TensorFlow for LSTM and RLlib for RL), and its versatility allows us to manage data processing, model training, and database interactions in one environment.

4.7.2. Skills to Be Developed

4.7.2.1. Long Short-Term Memory (LSTM)

LSTM networks are central to the project, as they are responsible for understanding sequential data patterns from the historical dataset. Gaining experience in sequential modeling is essential for accurate memorization and revision predictions.

4.7.2.2. Reinforcement Learning (RL)

RL is a specialized area of machine learning essential for this project. The RL model is responsible for dynamically adjusting daily recitation plans based on real-time feedback, making it central to the project's goal of adaptive memorization.

4.7.2.3. Data Processing and Feature Engineering

The accuracy of the LSTM model and RL agent relies heavily on clean, well-structured data. Developing data processing skills is crucial for managing historical data and creating real-time performance tracking metrics.

4.7.2.4. Backend Development

To effectively manage data interactions and integrate models, backend development skills are essential for this project. Backend skills enable the creation of a powerful system for retrieving, updating, and managing data from the database, especially with the addition of new recitation data. The use of backend frameworks such as Flask or Django enables secure and efficient processing of data requests and form outputs while facilitating the development of user interfaces such as APIs for teachers to input or view student progress. Backend development skills help ensure that the database, models, and interface components work together seamlessly, maintaining data integrity and supporting the model's adaptability.

4.8. The Initial Design

4.8.1. Use Case Diagram

The use case diagram illustrates the various functionalities of the Quran Memorization Plan Generator system from the perspective of a user. The primary actor interacting with the system is the user, who can be a teacher, administrator, or supervisor overseeing students' memorization progress. Below is a detailed description of the key use cases:

1. **View Students' Plans:** The user can view personalized memorization plans for individual students. These plans include daily goals for memorization and revision, designed to adapt to each student's progress and retention capacity.

2. **Record Recitation Feedback:** During or after a recitation session, the user can record feedback on the student's performance. This feedback is critical for evaluating progress, such as accuracy, speed, and retention.

3. **Adjust Plans After Recitation in Real-Time:** The system uses the feedback recorded by the user to dynamically adjust the student's plan. Based on the student's performance, the system can increase or decrease daily goals, prioritize revision, or modify the pace of memorization.

4. **Generate Periodic Plans:** The system periodically updates memorization plans to reflect long-term performance trends. These updates ensure the plans remain relevant and aligned with the student's overall progress.

5. **Pause or Modify an Ongoing Plan:** The user can manually intervene to pause or modify a student's memorization plan. This feature is helpful in cases where the student requires a temporary break or specific adjustments that the automated system may not account for.

6. **Add New Student:** When a new student is added, the system generates an initial plan. If historical data is unavailable for the new student, the system retrieves average performance metrics from similar students to provide a personalized starting plan.

7. **Generate Initial Plans:** Initial plans for students are generated by the system based on either historical data (for returning students) or average performance data (for new students). These plans provide a foundation that can be further refined through real-time adjustments.

8. **Generate Plans Manually:** The user has the option to create plans manually for students who may require a custom approach. This feature provides flexibility for unique cases where automated plans are insufficient.

The diagram emphasizes the modular nature of the system, highlighting the dependencies and extensions between various functionalities. It ensures flexibility, adaptability, and a user-centric approach to Quran memorization.

Figure 4: Use Case Diagram

4.8.2. ER Diagram

The shown Entity-Relationship (ER) Diagram represents the database design for the system that manages student memorization and tracks recitations. The tables and relationships illustrate how user data, student profiles, Quranic verses, surahs, and recitation sessions are organized. The tables shown in the ER diagram are:

- **USERS**: Manages user authentication and basic profile information, including fields such as user_id, username, password_hash, email, and timestamps for record creation and updates.
- **STUDENTS**: Stores student-specific details like student_id, demographics (age, gender, nationality), contact information, and a link to the parent user via user_id.
- **STUDENTS.PLANS_INFO**: Tracks memorization progress and revision metrics for students, including memorized parts, ratings, pages/letters amounts, revision directions, and timestamps. Linked to students via student_id.
- **VERSES**: Contains details of individual Quranic verses, such as verse_id, surah_id, positional attributes (page number, order in Quran), technical metrics (letters count, verse difficulty), and relationships to surahs.
- **SURAHS**: Lists metadata about Quranic chapters, including surah_id, name, and total number of verses.
- **RECITATION_SESSION**: Logs recitation sessions for students, recording session type, date, verses recited (start_verse_id, end_verse_id), performance ratings, acceptance status, and counts of pages/letters. Linked to students via student_id.

The tables are interconnected through key fields (e.g., student_id links students to their plans and recitation sessions, surah_id connects verses to surahs). This structure supports tracking memorization progress, managing user accounts, and analyzing recitation performance.

-

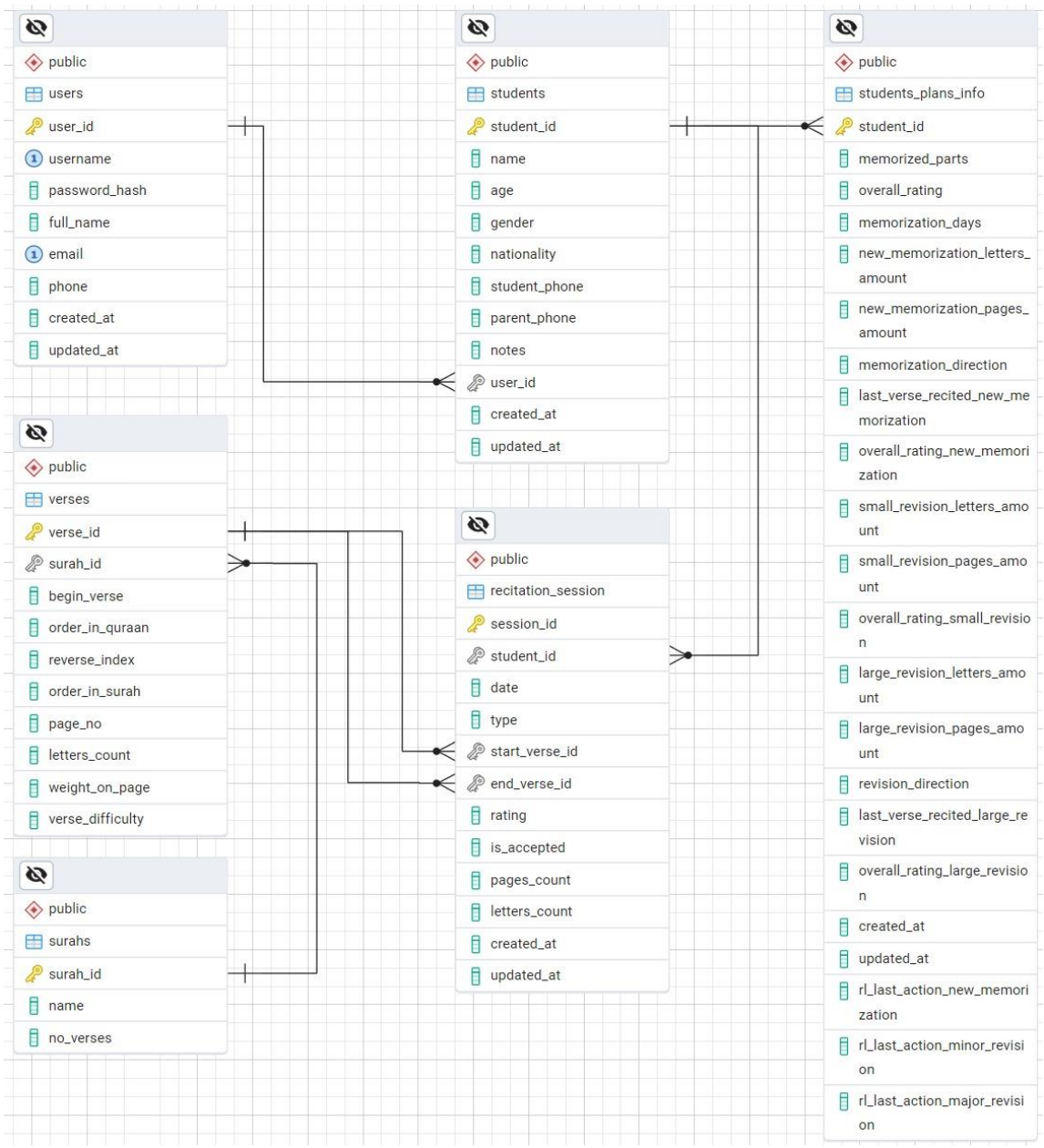

Figure 5: ER Diagram

4.8.3. Data Flow Diagram

A Data Flow Diagram (DFD) is a visual representation that shows how data moves through a system. It illustrates where data originates, how it flows between processes, how it's stored, and how it is used by the system. In this project, the DFD explains how student data is processed and used to create adaptive learning plans.

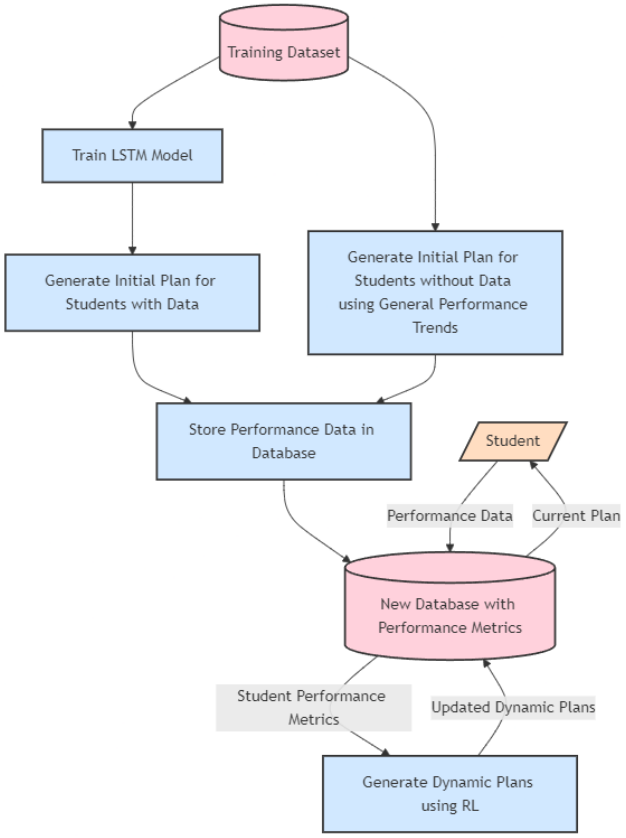

Figure 6: Data Flow Diagram

In the DFD above:

1. **Training Dataset:**

   This initial dataset contains historical data on students' recitation and memorization.

   - The LSTM Model is trained using this dataset to identify memorization patterns and generate initial plans for students with existing data.

- For students without historical data, general performance trends are calculated from this dataset to create an initial plan.

2. **Generate Initial Plans:**
    - For students with data, the LSTM model creates a plan based on their historical performance.
    - For new students, general trends are used to set the starting point.
3. **Store Performance Data in Database:**
    - Both initial plans and ongoing performance data are stored in the database, which collects real-time information on each student's progress.
4. **Database with Performance Metrics:**
    - The database receives current performance data from the students, including their progress on the memorization plan.
    - It stores metrics such as errors, rating, and recitation pace, which are used to dynamically adjust the student's learning plan.
5. **Generate Dynamic Plans using RL:**
    - The RL model accesses performance data from the database to create adaptive plans for each student.
    - Based on recent performance, the RL model makes real-time adjustments, providing students with an optimized plan that supports their individual learning pace and retention needs.

4.8.4. Flowchart

A flowchart is a visual representation of a process, showing each step in a sequence. It uses shapes like boxes, diamonds, and arrows to depict the flow of tasks and decisions within a system.

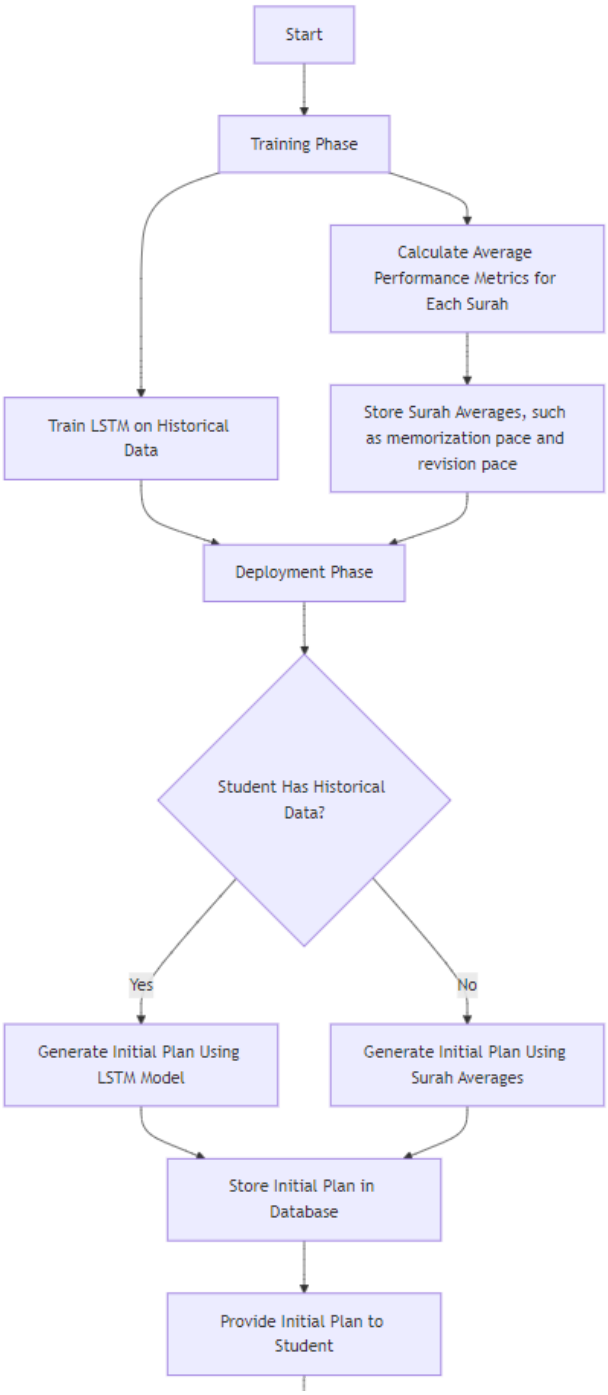

Figure 7: Flowchart (Part 1)

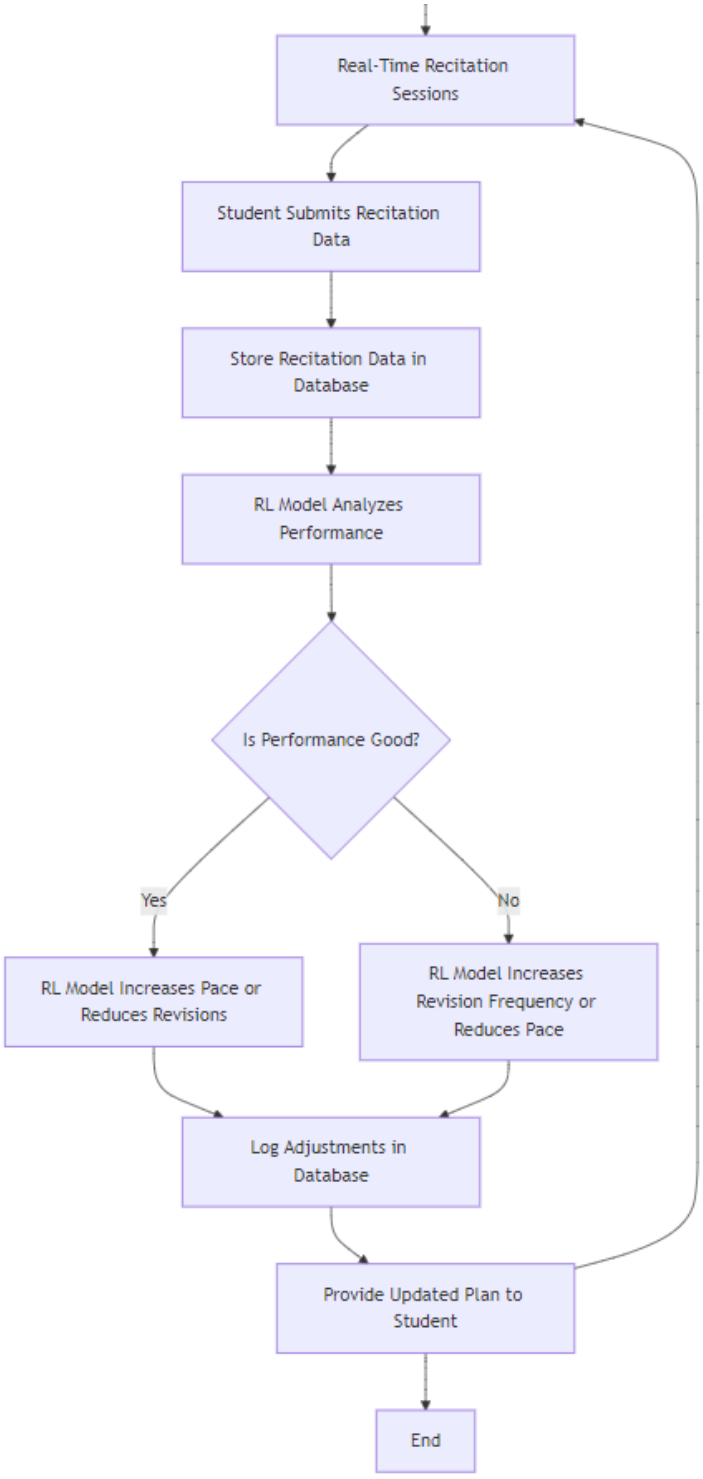

Figure 8: Flowchart (Part 2)

This flowchart illustrates the process for generating and adjusting personalized Quran memorization plans using LSTM and RL models.

1. **Training Phase**:
    - The process starts with the Training Phase, where the system trains an LSTM model on historical data to learn memorization patterns.
    - For students without historical data, average metrics for each Surah are calculated and stored.
2. **Deployment Phase**:
    - When a student needs an initial memorization plan, the system checks if they have historical data.
    - If historical data exists, the LSTM Model generates a plan. If not, the system uses Surah Averages to create the plan.
    - The initial plan is then stored in the database and provided to the student.
3. **Real-Time Recitation Sessions**:
    - During each recitation session, the student submits their performance data, which is stored in the database.
    - The RL Model then analyzes this data to evaluate if the student's performance is progressing well.
4. **Performance Evaluation and Adjustment**:
    - If the student performs well, the RL model may increase the pace or reduce revision frequency.
    - If performance needs improvement, the model may decrease the pace or increase revision sessions.
    - Adjustments are logged into the database, and an updated plan is provided to the student, ensuring it adapts to their needs in real time.

4.8.5. Sequence Diagram

A sequence diagram is a type of interaction diagram that shows how processes operate with each other and in what order. It focuses on the sequence of messages exchanged between components in a system to accomplish a particular function. Sequence diagrams help understand the flow of interactions over time, especially in complex systems.

4.8.5.1. Add New Student

This sequence diagram illustrates the process of adding a new student to the system. When a user submits the details of a new student, the system validates and stores the information in the database. After confirming successful storage, the system initializes a default learning plan for the student and stores it in the database. Finally, the system notifies the user that the new student has been added successfully. This workflow ensures that new students are seamlessly integrated into the system with an initial plan that can be customized further.

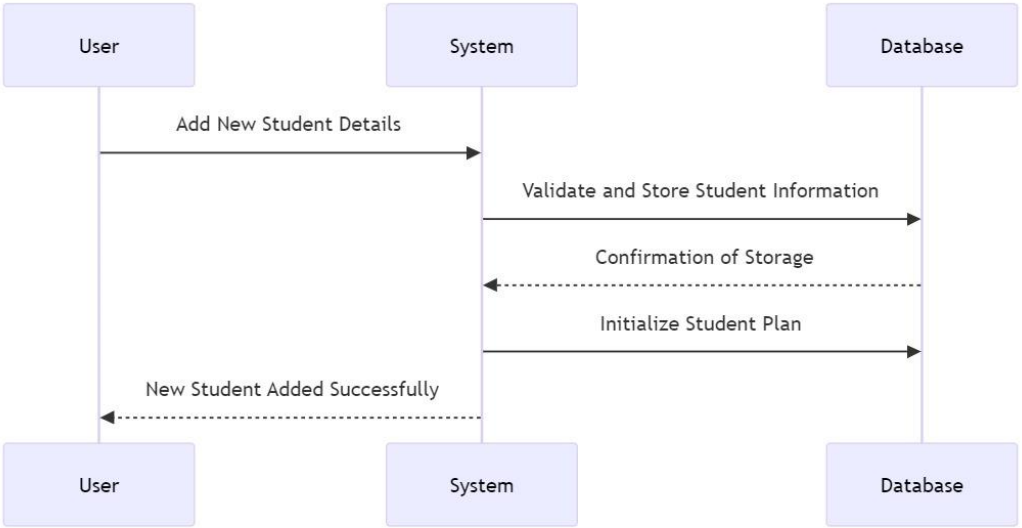

Figure 9: Sequence Diagram - Add New Student

4.8.5.2. Request Initial Plan

This sequence diagram explains how the system generates a personalized plan for a user. When a user requests an initial plan, the system first queries the database for any available historical performance data. If historical data exists, it retrieves this information and uses an advanced Long Short-Term Memory (LSTM) model to generate a personalized plan. If no historical data is available, the system relies on cluster-based averages retrieved from the database to create a plan using a predefined algorithm. Regardless of the method, the generated plan is stored in the database for future use. The user is then presented with their initial plan. This workflow ensures that the system can provide tailored plans, whether the user is new or has a performance history.

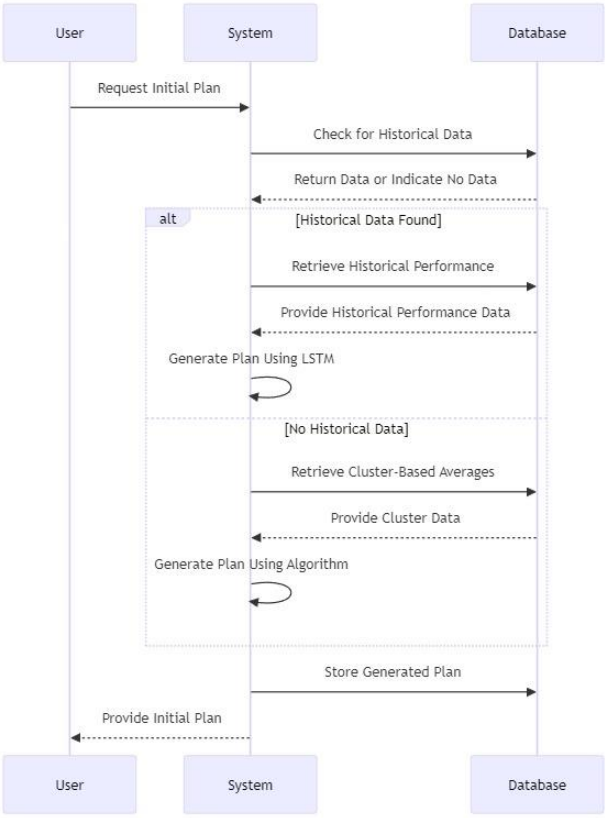

Figure 10: Sequence Diagram - Request Initial Plan

4.8.5.3. Record Recitation Feedback

This sequence diagram focuses on how the system incorporates user feedback to enhance their learning plan. Users submit recitation data, which the system stores in the database. The system then retrieves both historical and current performance data and sends it to a reinforcement learning (RL) model for analysis. The RL model adjusts the memorization plan based on this analysis and returns the updated plan to the system. The system stores the updated plan in the database and notifies the user of the changes. This workflow highlights the adaptive nature of the system, enabling it to refine plans in real time based on user performance.

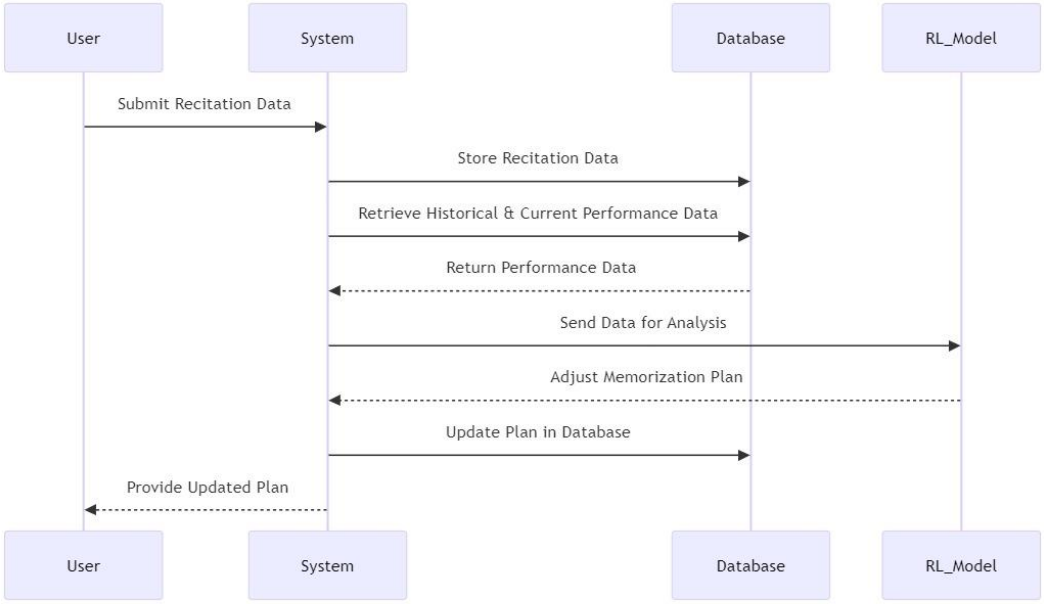

Figure 11: Sequence Diagram - Record Recitation Feedback

4.8.5.4. Pause or Modify an Ongoing Plan

This sequence diagram demonstrates the process by which a user can adjust an ongoing plan. It begins with the user submitting a request to pause or modify the plan. Upon receiving this request, the system retrieves the current plan details from the database. After obtaining the relevant data, the system processes the request to adjust the plan based on the user's input. The updated plan is then stored back in the database to ensure that changes are reflected in future actions. Finally, the system notifies the user that the requested modifications have been successfully applied. This workflow underscores the system's flexibility and its ability to cater to individual user needs dynamically.

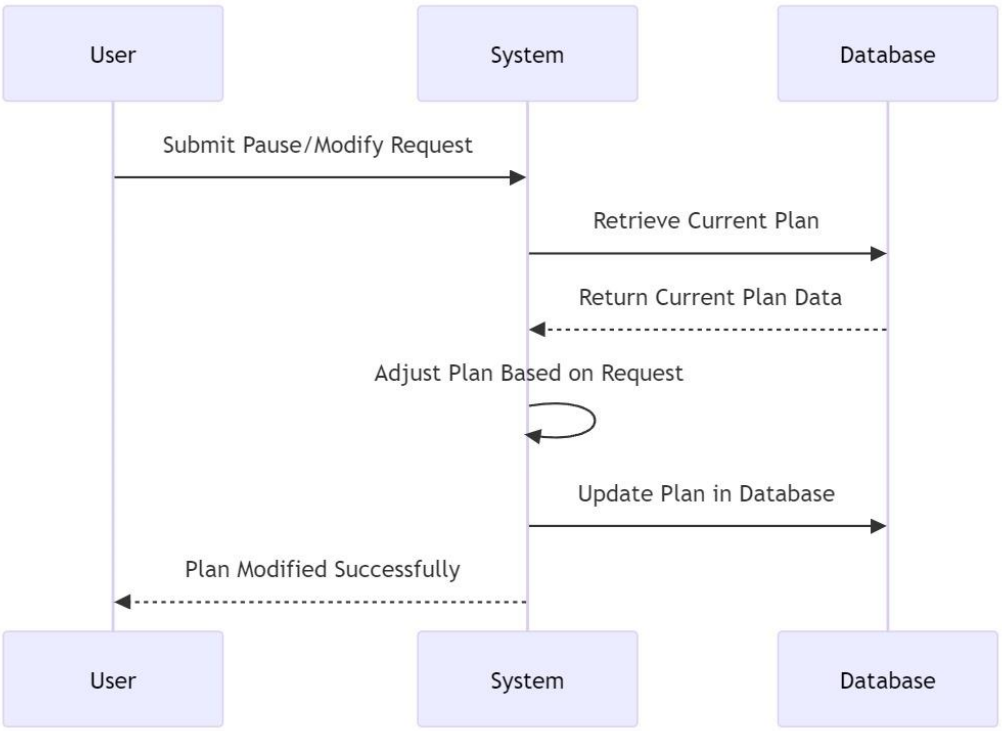

Figure 12: Sequence Diagram - Pause or Modify an Ongoing Plan

4.8.6. Prototype Design

4.8.6.1. Students' Page

On the Student's Page, the teacher can see the list of students assigned to them. They can review a student's plan, modify a student's information or plan amounts, delete a student, or add a new student. This is illustrated in Figure 13.

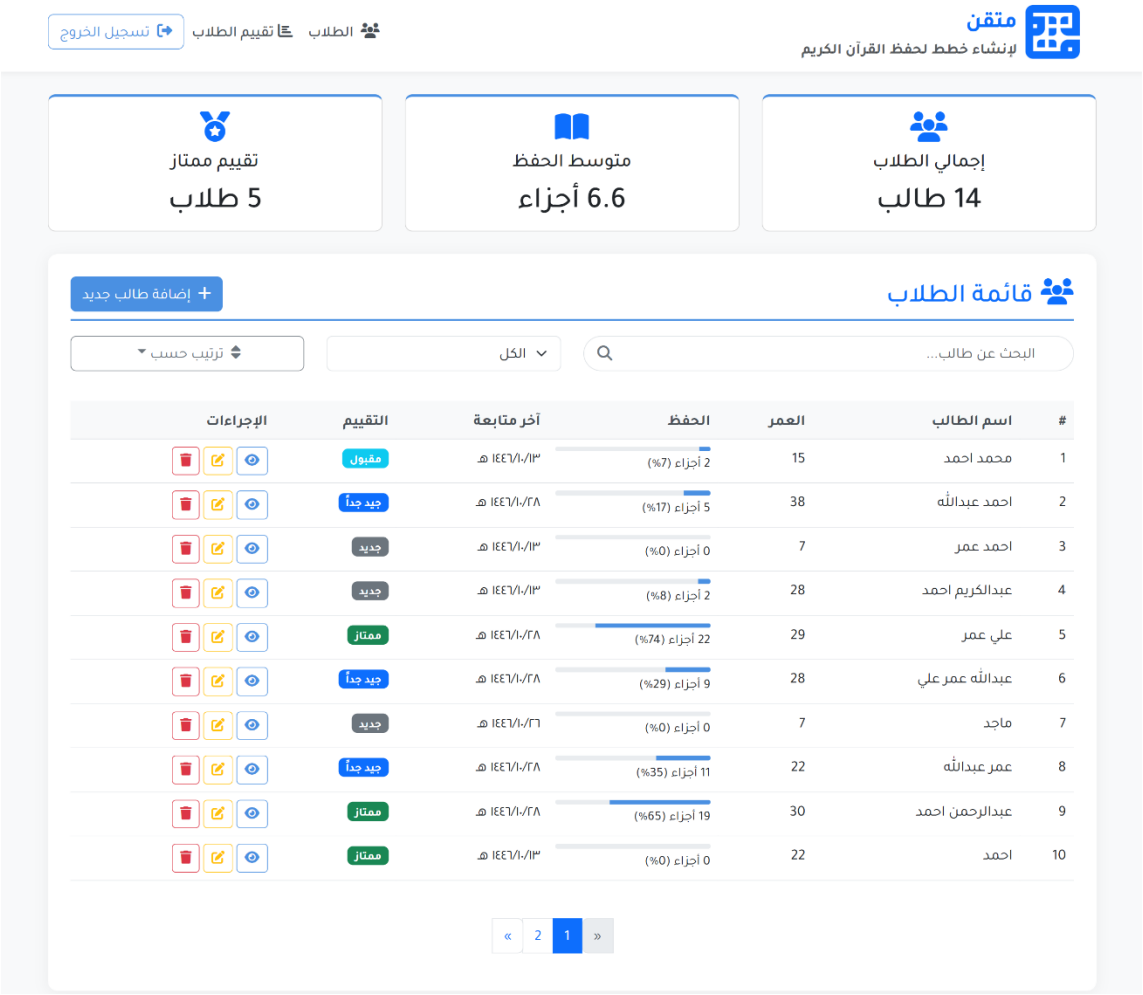

Figure 13: UI – Student's Page

4.8.6.2. Add New Student

On the Add New Student page, the teacher provides the student's details, such as name and gender, along with the initial period for the plan. If the student has memorized parts of the Quran or has previous plans, the teacher can include that information. The teacher can also set optional criteria for the new plan. This is shown in Figure 14.

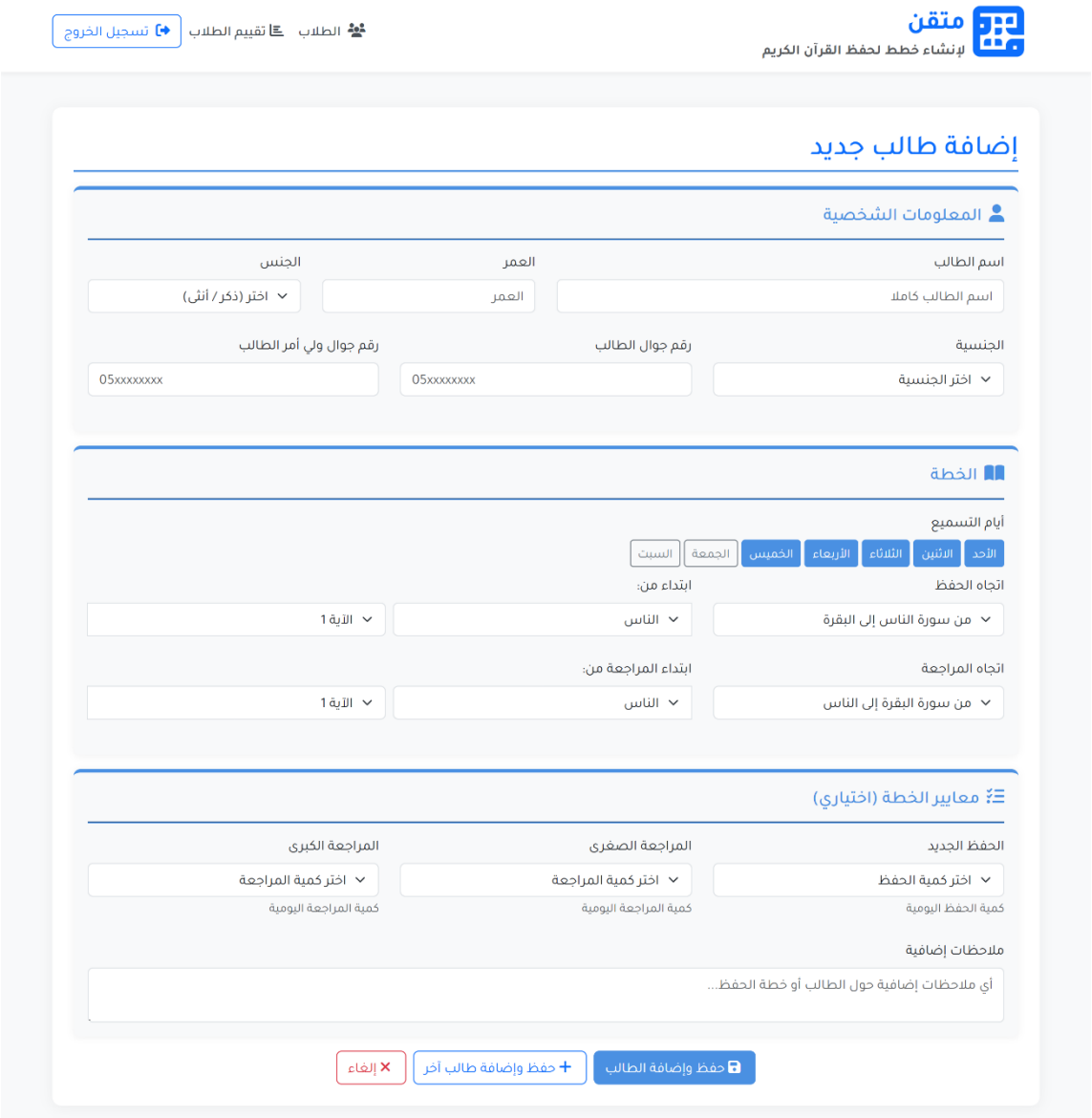

Figure 14: UI – Add a new student

4.8.6.3. Student's Weekly Plan

When the system generates a weekly plan for the student, the new plan is displayed with dates, and each session is colored according to its completion status; red for incomplete and green for complete. This is shown in Figure 15.

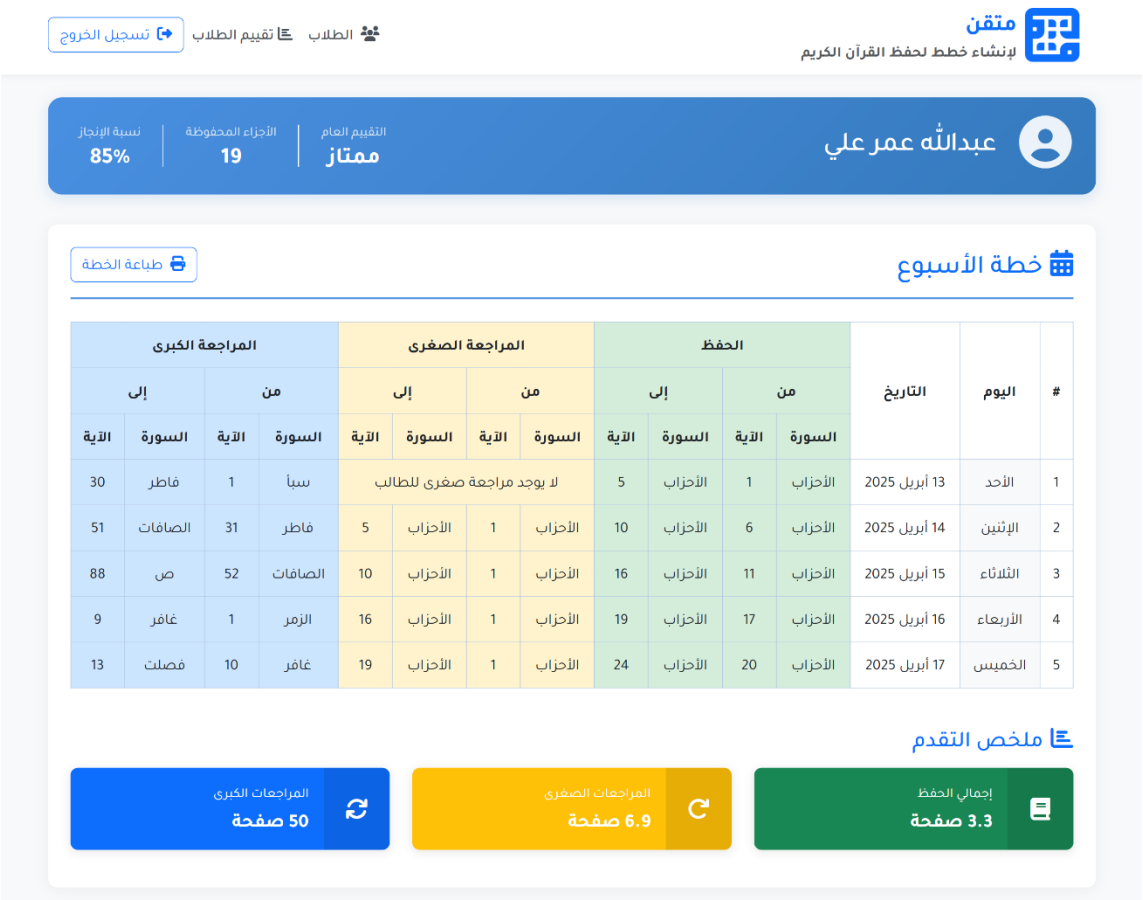

Figure 15: UI – Student's Weekly Plan

4.8.6.4. Evaluate Student's recitations

When the teacher views the student's plan, they can record the achievement and evaluation for each recitation. This information helps the model create a more tailored plan for the student in the future. This is shown in Figure 17.

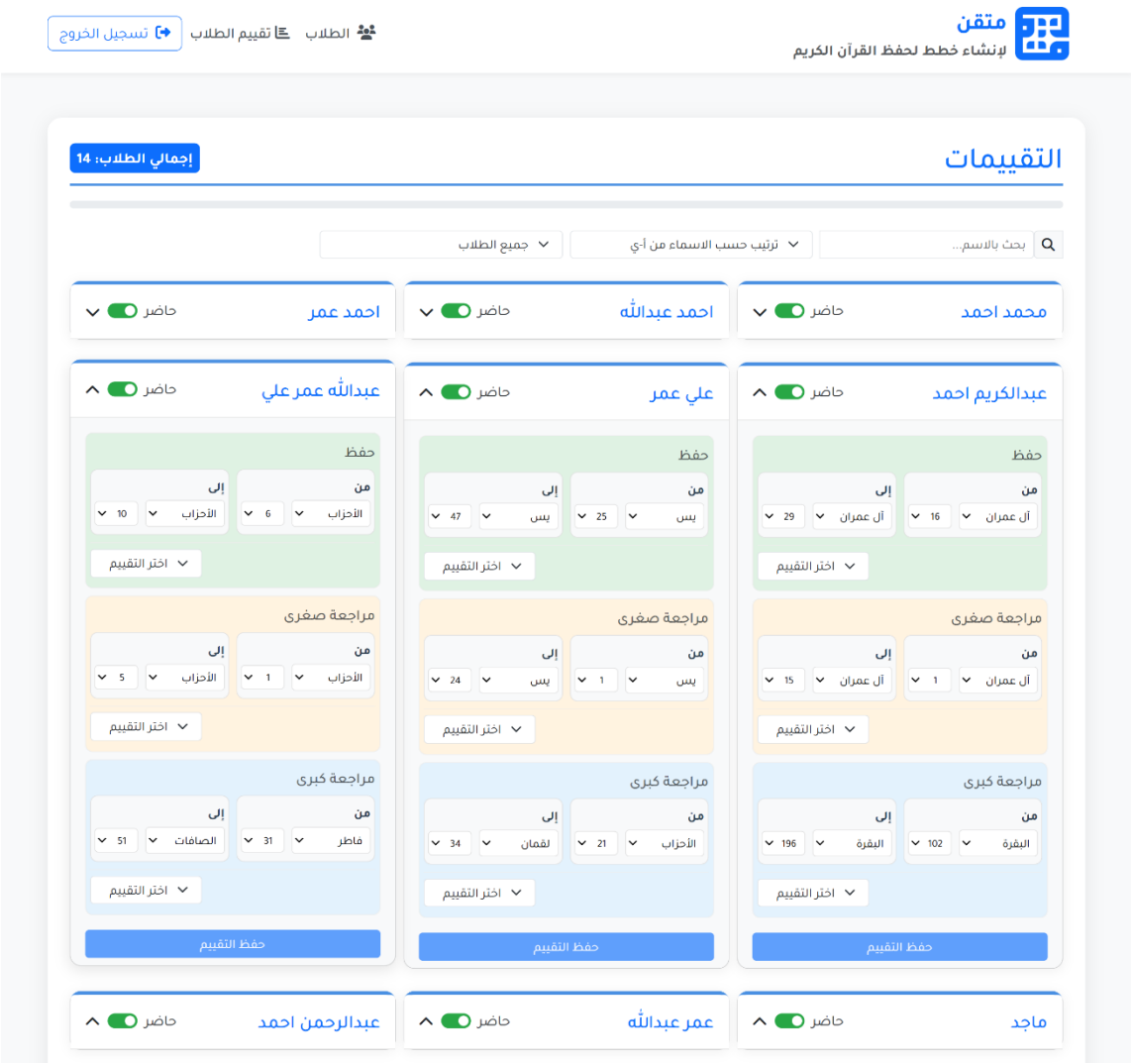

Figure 16: UI - Evaluate Student's recitations

CHAPTER 5

IMPLEMENTATION

5.1. Calculating Verse Difficulty Rates

To further enhance the personalization of memorization plans, we developed an algorithm to calculate the difficulty rates of Quranic verses. This step involved analyzing students' historical memorization data to identify patterns in how quickly or slowly verses were memorized. The resulting dataset, verse_difficulty.csv, provides a quantitative measure of the relative difficulty of each verse, which can be used to tailor memorization plans to individual students' needs. The goal was to assign a difficulty score to each Quranic verse based on students' memorization performance, which is how much of a memorization session a verse takes. Verses that took longer to memorize were considered more difficult, while those memorized quickly were considered easier.

The algorithm was implemented by processing recitation sessions. We focused on new memorization sessions (identified by pillar_id=1), as these sessions reflect the initial effort required to memorize verses. For each session, we retrieved the start_verse_id and end_verse_id to determine the range of verses memorized. Then, for each verse in the

	A	B
1	verse_id	average_difficulty
2	1	0.25439184
3	2	0.242977202
4	3	0.10635513
5	4	0.11013381
6	5	0.181880164
7	6	0.145407198
8	7	0.427967036
9	8	0.219097578
10	9	0.258710414
11	10	0.333137227
12	11	0.299677342
13	12	0.19130699
14	13	0.105480605
15	14	0.119127499
16	15	0.109996236
17	16	0.115227176
18	17	0.117317598
19	18	0.097912472

Figure 17: Verse

session, we calculated a difficulty ratio by dividing the verse's weight_on_page (a measure of its length) by the total calculated_pages_count for the session. This ratio represents the proportion of the session's workload attributed to that specific verse. A higher ratio indicates that the verse took up a larger portion of the session, suggesting it was more challenging to memorize. Following that, we aggregated difficulty ratios across all sessions for each verse and computed the average difficulty for that verse. For example, if a student memorized Surah Al-Baqarah, Verse 255 in a session with a total calculated_pages_count of 5, and the verse's weight_on_page is 0.5, the difficulty ratio for this verse would be 0.5 / 5 = 0.1. If multiple students took longer to memorize this verse, its average difficulty score would increase, reflecting its relative challenge.The final dataset, verse_difficulty.csv, contains two columns: verse_id and average_difficulty. This dataset will be used mainly to generate initial plans for students who have no historical data.

5.2. AI Models

5.2.1. Long Short-Term Memory (LSTM)

5.2.1.1. Cleaning the Dataset

A critical first step in the project was cleaning the dataset to ensure its reliability for training AI models. The raw data contained inconsistencies, outliers, and illogical entries that could skew model predictions. Below is a detailed breakdown of the steps taken.

1. **Dropping students with inconsistent memorization.**

Inconsistent memorization refers to illogical or erratic patterns in how students' progress through the Quran. These inconsistencies can arise from data entry errors, irregular learning habits, or other anomalies that disrupt the natural sequence of memorization. To ensure the dataset reflects realistic and structured learning behaviors, we implemented a systematic process to identify and remove students with inconsistent memorization patterns.

Students memorizing the Quran typically follow a logical order, either **forward** (starting from Surah Al-Fatiha and progressing to Surah An-Nas) or **reverse** (starting from Surah An-Nas and moving backward). Any deviation from this sequence, such as skipping Surahs or reversing direction mid-progress, was flagged as a violation. Students with more than 3 invalid entries were removed entirely. This threshold ensures that only students with consistent, logical memorization patterns are retained in the dataset.

By filtering out students with erratic memorization patterns, the dataset retains only those who follow structured and logical learning paths. This ensures that the LSTM model can accurately learn sequential dependencies.For example, a student who memorizes Surah Al-Baqarah (2) and then jumps to Surah An-Nisa (4) without completing Al-Imran (3) would be flagged. After 3 such violations, their data would be excluded from the dataset.

By retaining only consistent and logical memorization patterns, the AI models can generate accurate and adaptive plans tailored to each student's unique learning history. This step was crucial in preparing a high-quality dataset for training the AI models, ensuring that the generated memorization plans were effective and aligned with real-world learning practices.

2. Dropping the outliers

Outliers in the dataset refer to entries that deviate from realistic memorization patterns. These anomalies often arise due to data entry errors or exceptional cases that do not reflect typical student behavior. To ensure the dataset accurately represents genuine learning patterns, we implemented systematic checks to identify and remove such outliers. The distribution of the calculated_pages_count for each memorization session was as follows:

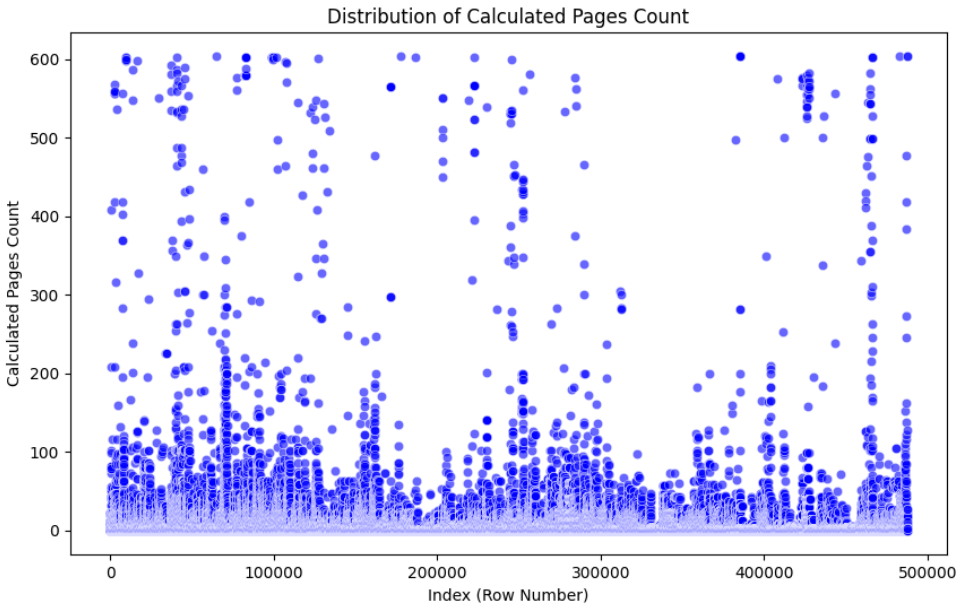

Figure 18: Number of pages in each record

The plot shows sessions with over 100, 200, and even 600 pages. These records were illogical and had to be dropped from the dataset.

To further investigate the entries for each type of session (new memorization and minor/major revisions), we visualized the calculated_pages_count for each of these types, and the distributions were as follows:

Figure 19: Number of pages in each new memorization

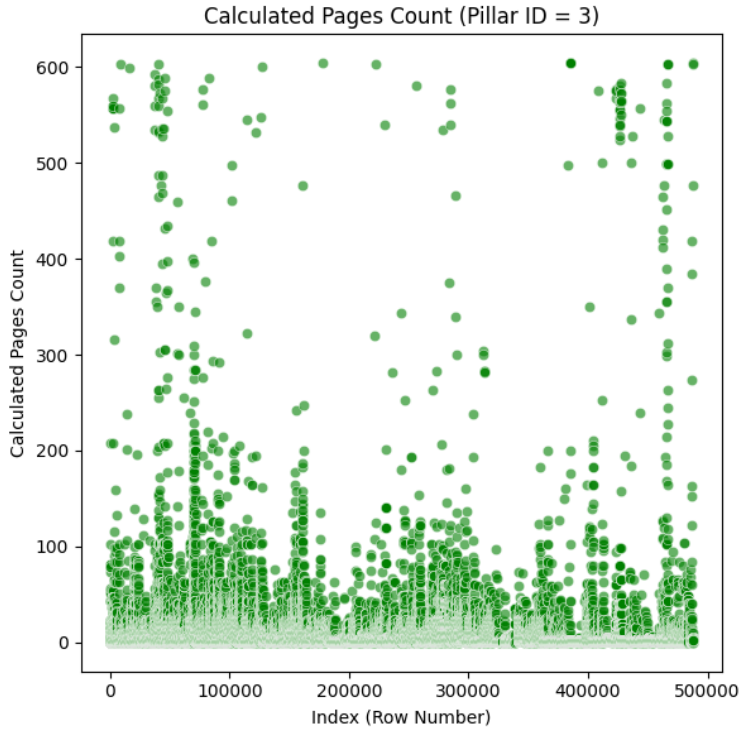

Figure 20: Number of pages in each minor revision

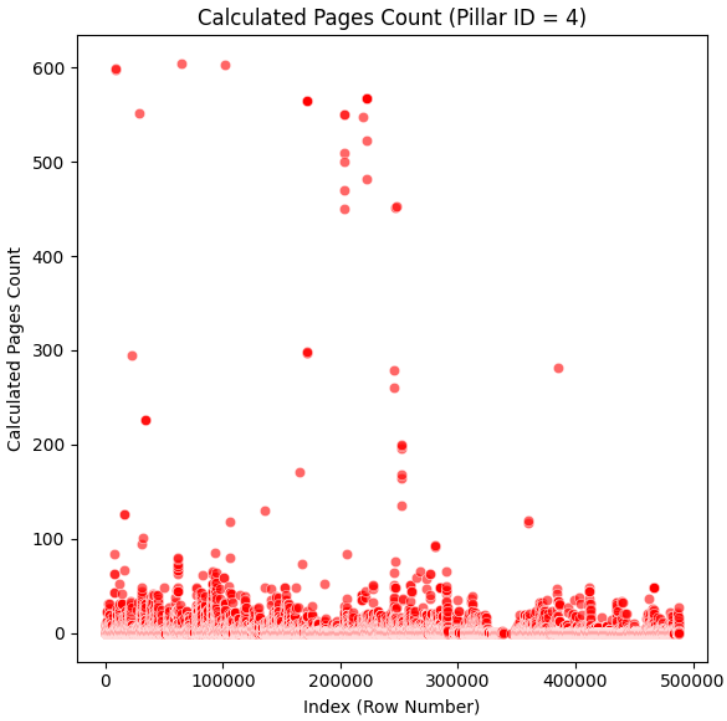

Figure 21: Number of pages in each major revision

These distributions show that illogical entries were present in each type of session. To drop these erroneous records, we set an appropriate threshold for each type of session. The thresholds are 20 pages for the new memorization, 50 for the minor revision, and 100 for the major revision. Since pages vary in length, we also tracked letters as a finer-grained metric. A threshold of 11,000 letters per day was set for new memorizations, 27,500 letters per day for minor revisions, and 55,000 letters per day for major revisions, calculated based on the average letters per page (~550 letters) multiplied by the threshold. Any student with 3 or more records that exceed the session type's threshold has been dropped, otherwise, only the record was dropped.

5.2.1.2. Training the LSTM Models

The Long Short-Term Memory (LSTM) model is a key component of the system, designed to analyze historical student performance and predict future memorization goals. It processes sequential data to identify patterns in how students memorize and revise Quranic verses. The training process involved preprocessing the cleaned dataset,

designing the model architecture, and training separate models for the three memorization session types.

1. Data Structuring

Before training the LSTM model, the dataset had to be structured into a format suitable for sequential learning. This involved several steps. The dataset was divided into three subsets based on the pillar_id column, which identifies the type of memorization activity. Each subset was processed separately to capture unique patterns in workload and pacing for each activity type. Features like letters_count and pages_count were scaled to a range of [0, 1] using **MinMaxScaler**. Normalization ensures that all features contribute equally to the model's learning process. For example, without scaling, letters_count (which can be in the thousands) might dominate pages_count (typically in the single digits), leading to biased predictions.

For each student, sequences of **9 consecutive sessions** were created to predict the **4th session's workload**. This approach allows the model to learn patterns over time. For example, If a student had sessions on Days 1, 2, 3, and 4, the input sequence would include data from Days 1–3, and the target output would be Day 4's letters_count and pages_count. Students with fewer than 3 sessions were excluded because the model requires sufficient historical data to identify patterns. The data was grouped by student_id to maintain individual learning trajectories. This ensures the model learns personalized patterns rather than generic averages across all students.

2. Model Architecture

The model uses a single LSTM layer with 64 units. Units (also called neurons) are the building blocks of neural networks. Each unit processes a portion of the input data and learns patterns over time. The input shape was set to (3, 2), where 3 represents the number of historical sessions and 2 represents the features (letters_count and pages_count). A Dense layer with 2 units and linear activation was added to predict the next session's letters_count and pages_count. The Dense layer connects all inputs to outputs, and linear

activation ensures the model outputs continuous values (e.g., 5.2 pages). The **Adam optimizer** was used to adjust the model's weights during training. Adam is an adaptive optimizer that automatically adjusts the learning rate, making it efficient for training deep learning models.

Mean Squared Error (MSE) was used to measure the difference between the predicted and actual workload values. MSE penalizes large errors more heavily, ensuring the model focuses on accurate predictions.

3. Training Process

Three separate models were trained, one for each pillar type. The models were trained for **50 epochs**, meaning the entire dataset was passed through the model 50 times. This allows the model to learn patterns gradually. Each training step used **16 sequences** (batch size). Smaller batches help the model generalize better by updating weights more frequently. After training, the models and scalers were saved for deployment. The **models were** saved as HDF5 files (model_pillar_1.h5, etc.) for future inference. HDF5 is a file format designed to store large datasets and models efficiently. **Scalers were** saved as pickle files (scaler_pillar_1.pkl, etc.) to normalize new input data during predictions. Pickle is a Python library for serializing objects, allowing the scalers to be reused.

The trained LSTM models form the foundation for generating baseline memorization plans. By analyzing historical trends, they predict achievable daily workloads for students, balancing new memorization with revision. When combined with the RL model (for real-time adjustments), this hybrid approach ensures that plans are both personalized and adaptive. This training phase marks a critical milestone in developing an AI-driven Quran memorization system, bridging data-driven insights with educational objectives.

5.2.1.3. Tools and Technologies

The implementation of the LSTM models uses Python, TensorFlow, and Keras, which are widely recognized for their flexibility, scalability, and ease of use in AI/ML projects. Python is the primary programming language for this project due to its extensive ecosystem of libraries and frameworks tailored for machine learning and data analysis. Python's simplicity and readability make it ideal for rapid prototyping and collaborative development. Libraries like Pandas and NumPy are used for data preprocessing, while TensorFlow and Keras provide the tools needed to build, train, and deploy the LSTM models.

TensorFlow[9], [10] is a powerful open-source machine learning framework developed by Google. It is particularly well-suited for deep learning tasks, such as training LSTM networks, due to its ability to handle large datasets and complex computations efficiently. TensorFlow's flexibility allows developers to customize models and experiment with different architectures, making it a natural choice for this project. Keras, which is integrated into TensorFlow, is a high-level neural network API that simplifies the process of building and training deep learning models. Keras provides a user-friendly interface for defining layers, compiling models, and training them with minimal code. This abstraction allows developers to focus on the logic of the model rather than the intricacies of low-level implementation.

5.2.2. Reinforcement Learning (RL) Model

In this project, RL is employed to dynamically adjust students' memorization and revision plans based on their performance history. The core objective is to personalize learning workloads in real-time, ensuring students are neither under-challenged nor overwhelmed. The Proximal Policy Optimization (PPO) algorithm serves as the foundation of the RL model, chosen for its balance of stability and efficiency in handling continuous action spaces. By integrating feedback from teacher-assigned session ratings, the model iteratively refines its strategy to optimize long-term student outcomes.

5.2.2.1. Model Architecture and Design Choices

1. **State Space Design**: The state space is a multi-dimensional representation of a student's learning context, capturing both historical performance and real-time dynamics. Key features include the student's overall performance rating and current workload targets. A Long Short-Term Memory (LSTM) model provides baseline predictions of future performance by analyzing sequential patterns in memorization history, such as success rates and performance variance. The state also integrates metrics like the percentage of the Quran memorized and session-specific ratings. To distinguish between session types—new memorization, minor revision, or major revision—a one-hot encoded vector is appended. This thorough design ensures the model accounts for both short-term trends and long-term progress, enabling accurate adjustments tailored to individual learning trajectories.

2. **Action Space Design**: The action space is defined as a continuous value within the range [-0.3, +0.3], representing percentage adjustments to the workload predicted by the LSTM model. This range allows the system to recommend incremental changes (±30%) to the assigned material. For example, a student demonstrating strong retention might receive a +20% increase in workload to foster growth, whereas a struggling learner could see a -15% reduction to alleviate pressure. By constraining adjustments within this bounded interval, the model

avoids drastic fluctuations that could disrupt learning consistency. This approach balances adaptability with stability, enabling dynamic fine-tuning of plans based on real-time performance feedback.

3. **Reward Function Design**: The reward function guides the model to prioritize the system's core educational objectives: steady progress and long-term retention of memorized content. It combines two key elements to shape the model's decisions. First, session ratings (scaled from 0 to 1) provide instant feedback on a student's performance in recent tasks. Second, a challenge bonus rewards the model when it assigns workloads that meet or slightly exceed the LSTM model's recommendations, encouraging students to stretch their abilities. To prevent complacency, the function penalizes the model if it reduces workloads unnecessarily for high-performing students, ensuring they remain adequately challenged. This balanced approach ensures the model optimizes for immediate success while fostering gradual skill improvement. By promoting incremental difficulty increases and discouraging overly cautious adjustments for capable learners, the system supports resilience and deeper mastery, aligning with the platform's mission to enable lasting Quranic memorization.

Together, these design choices create a reinforcement learning framework capable of dynamically adapting to individual student needs. The state space provides a comprehensive snapshot of the learner's context, the action space enables precise workload adjustments, and the reward function ensures alignment with pedagogical objectives. This architecture not only enhances personalization but also supports scalable, data-driven decision-making in Quranic education.

5.2.2.2. Training Process and Workflow

The workflow begins with constructing a specialized learning environment that encapsulates the students' recent recitation sessions, performance metrics, and progress

indicators. This environment is carefully engineered to represent the student's state through a multidimensional state space, capturing aspects such as average ratings, success rates, performance variance, predicted future performance, and normalized progress measures.

Central to the workflow is integrating the LSTM model, which forecasts the student's likely memorization capacity for upcoming sessions. This prediction is used as a baseline, upon which the RL agent proposes adjustments. The RL agent, trained using the Proximal Policy Optimization (PPO) algorithm, interacts with the environment by selecting actions corresponding to percentage adjustments in the recommended memorization amount. Each action is evaluated through a reward mechanism that balances the ambition of the recommendation with the actual outcomes observed in the student's subsequent performance, thus encouraging the agent to propose challenging and achievable plans.

The training process involves iteratively simulating student sessions within the environment. At each step, the agent observes the current state, selects an action, and receives feedback as a reward. The environment dynamically updates to reflect the consequences of the agent's decisions, ensuring that the learning process remains closely tied to real-world educational outcomes. The agent's policy is refined over multiple episodes, gradually improving its ability to personalize memorization plans to maximize student engagement and success.

This workflow is further enhanced by mechanisms for saving and reloading trained models, enabling continuous improvement as more student data becomes available. The design choices reflect a commitment to technical strictness and educational significance, ensuring that the RL model not only optimizes for quantitative performance metrics but also aligns with the broader pedagogical goals of adaptive, student-centered learning. Through this approach, the system exemplifies the application of advanced machine learning techniques to the domain of personalized education, offering a scalable and data-driven solution for Quran memorization planning.

5.2.2.3. Libraries and Tools

1. **Proximal Policy Optimization (PPO)** is a reinforcement learning algorithm designed to strike a balance between stability and sample efficiency. Developed by OpenAI, PPO updates policies in a way that avoids drastic changes by clipping the objective function, ensuring training remains reliable even in complex environments. In this project, PPO was chosen for its ability to handle continuous action spaces while maintaining robust performance across diverse student profiles. Its clipped surrogate objective minimizes harmful policy updates, making it ideal for real-world applications where safety and consistency are critical. PPO's implementation via Stable Baselines3 provides out-of-the-box support for parallelized training and hyperparameter tuning, streamlining deployment [11].

2. **Stable Baselines3**: The Proximal Policy Optimization (PPO) algorithm is implemented using the Stable Baselines3 library, a high-quality, modular reinforcement learning framework built on PyTorch. This library provides pre-configured, production-ready RL algorithms, ensuring reliable training pipelines and reproducibility. Its compatibility with Gym environments simplifies integration, while built-in features like automatic hyperparameter tuning and GPU acceleration streamline the development process [12].

3. **Gym**: The Gym library is an open-source toolkit developed by OpenAI for designing, implementing, and benchmarking reinforcement learning algorithms. It provides a standardized interface to create environments where agents learn by interacting with simulated or real-world systems through defined states, actions, and rewards. In this project, Gym was chosen to structure the interaction between the RL agent and the student learning environment. By encapsulating database queries, state representation, and reward calculation within a custom Gym environment, the system ensures consistency with widely adopted RL practices. This standardization simplifies integration with libraries like Stable Baselines3, enabling efficient policy updates and compatibility with pre-built algorithms.

Gym’s modular design also allows seamless testing of alternative RL approaches without disrupting core logic [13].

4. **SQLAlchemy**: Database interactions are managed via SQLAlchemy, a Python SQL toolkit that provides ORM (Object-Relational Mapping) capabilities. This library facilitates seamless querying of student profiles and session histories from PostgreSQL, while transactional updates ensure atomicity and consistency when modifying workload plans [14].

5.3. Database

The database is designed to support a Quran memorization and revision platform, leveraging PostgreSQL for scalability, relational integrity, and adaptive functionality. It focuses on tracking users, students, Quranic content (surahs and verses), recitation sessions, and personalized memorization plans, with reinforcement learning (RL) integration to dynamically adjust plans based on user progress. Below is a detailed overview.

5.3.1. Tables Details

5.3.1.1. users

The users table stores account information for system users, such as instructors. Each user can manage multiple students.

Columns:

- **user_id**: Primary key. Uniquely identifies each user.
- **username**: Unique login name for the user.
- **password_hash**: Securely stores the user's password in hashed form.
- **full_name**: The user's full name.
- **email**: User's email address (unique, optional).

- **phone**: User's phone number (optional).
- **created_at**: Timestamp for when the user account was created.
- **updated_at**: Timestamp for the last update to the user account.

Relationships:

- **students**: One-to-many. Links to all students managed by this user.

5.3.1.2. students

The students table contains personal and demographic information about each student, including contact details and the user (instructor or guardian) responsible for them.

Columns:

- **student_id**: Primary key. Uniquely identifies each student.
- **name**: Full name of the student.
- **age**: Student's age.
- **gender**: Student's gender. Must be 'M' (male) or 'F' (female), enforced by a check constraint.
- **nationality**: Student's nationality.
- **student_phone**: Student's phone number (optional).
- **parent_phone**: Parent or guardian's phone number (optional).
- **notes**: Additional notes about the student (optional).
- **user_id**: Foreign key to users.user_id. Indicates which user manages this student. If the user is deleted, this is set to null.
- **created_at**: Timestamp for when the student record was created.
- **updated_at**: Timestamp for the last update to the student record.

Relationships:

- **recitations**: One-to-many. All recitation sessions for this student.
- **students_plans_info**: One-to-one. Contains detailed information about the memorization and revision plans for this student.

5.3.1.3. surahs

The surahs table represents the chapters of the Quran. Each surah contains metadata and is linked to its verses.

Columns:

- **surah_id**: Primary key. Uniquely identifies each surah (chapter).
- **name**: Name of the surah.
- **no_verses**: Number of verses in the surah.

Relationships:

- **verses**: One-to-many. All verses belonging to this surah.

5.3.1.4. verses

The verses table contains detailed information about each verse in the Quran, supporting both forward and reverse memorization strategies.

Columns:

- **verse_id**: Primary key. Uniquely identifies each verse.
- **surah_id**: Foreign key to surahs.surah_id. Indicates which surah this verse belongs to.
- **begin_verse**: The beginning text of the verse.

- **order_in_quraan**: The verse's sequential order in the entire Quran, from Al-Fatiha to An-Nas (used for forward memorization). For example, the first verse of Al-Fatiha has order_in_quraan = 1, and the last verse of An-Nas has order_in_quraan = 6236.
- **reverse_index**: The verse's reverse sequential order, counting from the end of the Quran, from An-Nas to Al-Fatiha (used for reverse memorization). For example, the first verse of An-Nas has reverse_index = 1, and the last verse of Al-Fatiha has reverse_index = 6236.
- **order_in_surah**: The verse's sequential order within its surah.
- **page_no**: The page number in the Quran where this verse appears.
- **letters_count**: The number of letters in the verse.
- **weight_on_page**: A float between 0 and 1 representing how much of the page the verse occupies, based on its letter count (e.g., 0.5 = half page).
- **verse_difficulty**: Numeric value representing the difficulty of memorizing this verse, calculated by a custom algorithm.

5.3.1.5. recitation_session

The recitation_session table logs each recitation session for a student, including what was recited, the number of letters and pages, the date of the session, and how well the student performed.

Columns:

- **session_id**: Primary key. Uniquely identifies each session.
- **student_id**: Foreign key to students.student_id. Indicates which student this session belongs to.
- **date**: Date of the recitation session.

- **type**: Type of session. Must be one of 'New_Memorization', 'Minor_Revision', or 'Major_Revision' (enforced by a check constraint).
- **start_verse_id**: Foreign key to verses.verse_id. The first verse recited in this session.
- **end_verse_id**: Foreign key to verses.verse_id. The last verse recited in this session.
- **rating**: Numeric rating of the session's performance (optional).
- **is_accepted**: Boolean indicating if the session was accepted (optional).
- **pages_count**: Number of pages recited in this session (optional).
- **letters_count**: Number of letters recited in this session.
- **created_at**: Timestamp for when the session was created.
- **updated_at**: Timestamp for the last update to the session.

Relationships:

- **student**: Many-to-one. Links back to the student who did the session.

5.3.1.6. students_plans_info

The students_plans_info table holds all information related to the student's memorization and revision plans, including direction, overall rating, amounts, and reinforcement learning adjustments.

Columns:

- **student_id**: Primary key and foreign key to students.student_id. Uniquely identifies the student and links to their plan.
- **memorized_parts**: Number of Quran parts (juz) memorized by the student (optional).
- **overall_rating**: Student's overall performance rating (optional).

- **memorization_days**: Number of days per week allocated for memorization (default is 5).
- **new_memorization_letters_amount**: Target number of letters to memorize in each new session (optional).
- **new_memorization_pages_amount**: Target number of pages to memorize in each new session (optional).
- **memorization_direction**: Boolean indicating memorization direction. True = forward (Al-Fatiha → An-Nas), False = reverse (An-Nas → Al-Fatiha).
- **last_verse_recited_new_memorization**: verse_id of the last verse recited in a new memorization session.
- **overall_rating_new_memorization**: Performance rating for new memorization sessions (optional).
- **small_revision_letters_amount**: Target number of letters for minor revision sessions (optional).
- **small_revision_pages_amount**: Target number of pages for minor revision sessions (optional).
- **overall_rating_small_revision**: Performance rating for minor revision sessions (optional).
- **large_revision_letters_amount**: Target number of letters for major revision sessions (optional).
- **large_revision_pages_amount**: Target number of pages for major revision sessions (optional).
- **revision_direction**: Boolean indicating revision direction. True = forward, False = reverse.
- **last_verse_recited_large_revision**: verse_id of the last verse recited in major revision (optional).

- **overall_rating_large_revision**: Performance rating for major revision sessions (optional).
- **rl_last_action_new_memorization**: Most recent RL adjustment for new memorization. A float (typically 0–1) indicating the effect of the RL system's latest action.
- **rl_last_action_minor_revision**: Most recent RL adjustment for the last minor revision session (optional).
- **rl_last_action_major_revision**: Most recent RL adjustment for the last major revision session (optional).
- **created_at**: Timestamp for when the plan was created.
- **updated_at**: Timestamp for the last update to the plan.

Relationships:

- one-to-one link to students.

5.3.2. Key Features

- **Data Integrity:** The schema enforces data integrity through check constraints (such as ensuring gender is either 'M' or 'F', and session type is valid) and foreign key relationships. Timestamp columns (created_at, updated_at) are included in all major tables to provide automated audit trails and track changes over time.
- **Normalization:** The database is normalized to reduce redundancy. For example, each verse is stored only once in the verses table and referenced by ID in related tables such as recitation_session. Similarly, user and student information is separated, with clear foreign key relationships.
- **Performance Optimization:** Relationships between tables (such as one-to-many and one-to-one) are explicitly defined to ensure efficient data retrieval, management, and cascading cleanup (e.g., deleting a student also removes their associated plans and sessions if needed).

5.3.3. Workflow Integration

The schema is designed to support the full workflow of Quran memorization and revision:

- **Granular Progress Tracking:** Progress is tracked at the verse level, allowing for detailed monitoring of what each student has memorized or revised.
- **Dynamic Planning:** Instructors can assign recitation sessions linked to specific Quranic content. The students_plans_info table enables dynamic adaptation of memorization and revision plans based on each student's performance, direction (forward or reverse), and reinforcement learning adjustments.
- **Personalized Learning Paths:** The structure supports personalized learning paths, with plans that adapt to the student's pace and ability, and session logs that record both the amount (in letters/pages) and quality of recitation.
- **Scalable Reporting:** Centralizing user-student relationships and Quranic metadata enables scalable reporting and analytics, supporting both administrative oversight and pedagogical refinement.
- **Streamlined Administration:** By organizing users, students, Quranic content, and plans in a relational structure, the system streamlines both administrative and instructional workflows, making it easier to manage large numbers of students and sessions.

This schema provides a robust foundation for a data-driven, adaptive Quran memorization platform, ensuring consistency, flexibility, and scalability across all core operations.

5.4. Frontend

The frontend of the Quran Memorization Plan Generator is designed to deliver a user-friendly and efficient experience for educators, emphasizing clarity and adaptability. Built using a modular approach, the system ensures consistency across all pages through reusable components such as navigation bars, styling elements, and interactive UI features. The Bootstrap 5 [15] framework forms the backbone of the responsive layout,

enabling seamless access across desktops, tablets, and mobile devices. Special attention is given to Arabic language compatibility, with full Right-to-Left (RTL) layout support and the use of the Tajawal font family for enhanced readability. Modern UI components—such as searchable tables, color-coded progress indicators, and dynamic forms—are integrated to streamline workflows and reduce cognitive load for users.

5.4.1. Key Pages and Functionalities

The system contains four core pages tailored to support Quran memorization management:

1. **Home Page (Dashboard and Student List):** Serves as a central hub, displaying statistics like total students, average memorization progress, and high-performing students. A searchable, filterable student table allows efficient record management with CRUD (Create, Read, Update, Delete) functionalities.

2. **Add Student Page:** Features a multi-section form for capturing personal details, previous memorization details, and optional criteria for plan generation using the teacher's input. Input validation ensures data integrity, while responsive design enables accessibility on mobile devices.

3. **Student Plan Page:** Provides an interactive interface to view memorization schedules. Goals and recitation targets are displayed on cards, with dynamic adjustments based on selected days.

4. **Student Evaluation Page:** Facilitates structured assessments through memorization, minor revision, and major revision tracking. Teachers can grade students using card-based profiles, with search, filter, and sorting tools to prioritize evaluations.

5.4.2. Tools and Technologies

The frontend uses HTML5 and CSS3 for semantic structure and styling, ensuring compliance with modern web standards. Bootstrap 5 enhances responsiveness and simplifies UI development through pre-built components, while JavaScript and jQuery enable dynamic interactions such as real-time form validation and AJAX-based updates. Font Awesome icons and the Tajawal font family (via Google Fonts) elevate visual appeal and readability, particularly for Arabic text. These technologies collectively ensure a smooth, intuitive experience across diverse devices and user needs.

5.4.3. Design and Functional Highlights

5.4.3.1. User-Centered Design Principles

Clarity and accessibility drive the design, featuring an RTL layout aligned with Arabic reading patterns and the Tajawal font for legibility. Modular layouts using Bootstrap's grid system organize content, while persistent navigation bars and contextual headers guide users through workflows. Visual hierarchy—via accent colors, card-based layouts, and button sizing—directs focus to key actions like form submissions.

5.4.3.2. Responsive and Adaptive Design

The interface adapts seamlessly across devices using media queries and Bootstrap utilities. On mobile, multi-column layouts collapse vertically, navigation condenses into hamburger menus, and tables become scrollable. Touch-friendly controls and dynamic font scaling ensure usability on-the-go without sacrificing functionality.

5.4.3.3. Pedagogical Integration and Aesthetic Consistency

Design choices are guided by educational best practices. Smart defaults, pre-populated Surah dropdowns, and dynamic filters minimize manual input. Progress indicators, such as color-coded charts, offer real-time feedback to inform teaching strategies. Consistent

micro-interactions, such as loading spinners and success notifications, reinforcing brand identity and user confidence.

5.5. Backend

5.5.1. Backend Architecture Overview

The backend forms the operational core of the Quran Memorization Plan Generation system, orchestrating critical workflows to ensure seamless interaction between users, students, and Quranic content. Designed to automate student management, generate personalized memorization plans, track recitation sessions, and evaluate performance, the backend leverages adaptive algorithms and real-time data processing to tailor learning experiences. By continuously monitoring student progress, integrating AI-driven recommendations, and applying pedagogical best practices, the system optimizes memorization and revision schedules for sustainable progress.

Built with scalability and data integrity as foundational principles, the architecture employs robust transaction management, error handling, and modular design. This structure ensures maintainability while enabling future enhancements with minimal disruption. The backend combines layered architecture, MVC (Model-View-Controller) principles, and a service-oriented design, ensuring clear separation of concerns and efficient data flow. At its core, the system is structured around three key components: modular services that encapsulate business logic, route handlers that manage API communication, and database models that define data structure and interactions. This division of responsibilities promotes testability, adaptability, and maintainability, allowing the system to evolve alongside user needs and technical requirements.

5.5.2. Data Flow and Request Handling

When a request is received from the frontend, it is first processed by route handlers in the Routes module, which act as the system’s entry point. These handlers validate incoming data, manage API endpoints, and route requests to the appropriate services. For

instance, the Students route manages operations such as retrieving student lists, adding new records, and updating details, serving frontend pages like the students management and the add student interfaces. Similarly, the RecitationSession route handles recitation session creation, evaluation updates, and data retrieval, supporting pages like the Student Plan and Evaluation interfaces. The weekly plan generation route provides an endpoint for generating a student's weekly memorization and revision plan, enabling the frontend to trigger advanced plan creation logic and retrieve tailored schedules for individual students.

Once routed, the request is processed by dedicated service classes, which encapsulate the system's core business logic. Services interact with database models via SQLAlchemy ORM, abstracting direct database operations. For example, the Plan Generation Service dynamically constructs weekly memorization plans by analyzing student history, preferences, and AI recommendations, while the Manage Recitation Evaluation Service adjusts future plans based on recitation performance. After processing, results are returned to the route handler, formatted, and sent back to the frontend. This layered approach—spanning route handling, service execution, and data modeling—ensures efficient data flow, clean code organization, and robust error handling, while maintaining alignment with the system's architectural principles.

5.5.3. Backend Services Overview

The backend's functionality is driven by specialized services, each addressing a unique aspect of the system. Below is a detailed breakdown of their roles and workflows:

5.5.3.1. Student Service

The Student Service class serves as the central interface for managing all student-related operations within the backend. It ensures that the creation, retrieval, updating, and deletion of student records are performed consistently and securely. These operations are built upon the students table, which forms the foundation of the system's data model by storing essential information about each learner—such as their name, age, gender, nationality, and contact details. Importantly, Student Service does not operate in isolation;

it is tightly integrated with additional backend services, including those responsible for managing memorization plans. When a new student is added, the service automatically coordinates the initialization of a personalized memorization plan, ensuring that each learner's progress and study path are tracked from the outset. This seamless integration between basic student data and individualized planning reflects the system's holistic approach to supporting both administrative efficiency and adaptive, student-centered learning experiences. By centralizing these operations, the backend not only streamlines routine management tasks but also lays the groundwork for advanced features such as dynamic plan generation and performance analytics, ultimately enhancing the educational value and scalability of the platform.

5.5.3.2. Student Plan Info Service

The Student Plan Info Service class is tightly linked to the Student Plans Information table, which contains detailed records for each student's memorization plan, including columns such as memorization and revision directions, the number of memorization days, the last verse recited in both new memorization and large revision, as well as metrics for the amount of new and revision memorization in letters and pages, overall ratings, and reinforcement learning adjustments. This service manages the creation, retrieval, updating, and deletion of these records, ensuring that each student's plan is accurately maintained and dynamically updated as they progress. In addition to basic operations, the class offers specialized methods that support the dynamic needs of the system. For instance, it includes functionality to determine how many parts (juz) a student has memorized based on their current progress, to identify the last verse recited according to the memorization direction, and to compute the appropriate starting surah and verse for the next session. Through these mechanisms, the service supports a personalized and adaptive learning experience, ensuring that each student's plan reflects their actual progress and provides clear guidance for their next steps.

5.5.3.3. Surah Service

The Surah Service class is a core part of the backend responsible for managing and providing access to information about the surahs, or chapters, of the Quran. This service streamlines the retrieval of surah data, enabling the application to efficiently access details such as surah names, the number of verses in each surah, and other relevant metadata. By centralizing these operations, the Surah Service ensures that all components of the system can consistently and reliably obtain the surah information they require, whether for displaying content, generating memorization plans, or supporting user queries. Its design supports both comprehensive listings and targeted lookups, contributing to the overall robustness and maintainability of the backend. This approach not only simplifies the management of Quranic chapter data but also enhances the system's ability to adapt to various educational and administrative needs.

5.5.3.4. Verses Service

The Verses Service class serves as a fundamental component in managing and accessing Quranic verse data within the backend architecture. This service provides essential capabilities for retrieving and navigating through verses in both forward and reverse directions, supporting the unique requirements of Quranic memorization. It enables direct access to individual verses, facilitates the location of specific verses within surahs, and supports sequential navigation as well as access by positional index within the Quran. A particularly notable feature is its directional functionality, which allows navigation to previous or next verses in either the traditional forward order (from Al-Fatiha to An-Nas) or in reverse order (from An-Nas to Al-Fatiha), thereby accommodating different memorization approaches. By centralizing these verse-related operations, the Verses Service class ensures consistent and efficient access to Quranic content throughout the application, supporting both traditional sequential memorization and reverse memorization methods. This versatility in verse navigation and retrieval is crucial for supporting various memorization strategies and maintaining the system's adaptability to different teaching methodologies.

5.5.3.5. Plan Generation Service

The Plan Generation Service class is responsible for constructing personalized weekly memorization and revision plans for each student, based on their current progress and preferences. To begin, the service gathers all relevant information for each type of recitation—whether it is new memorization, minor revision, or major revision—including the last verse recited, the direction of recitation, the amount to be covered (in letters or pages), and the specific days the student has chosen for their sessions. It also takes into account the starting date for the weekly plan.

For each day selected by the student, the service generates a tailored plan for every type of recitation. In the case of new memorization, the plan is primarily determined by the number of letters specified in the database, which is set by AI models to match the student's ability. If the number of letters is not available, the system uses the number of pages optionally provided by the user; if neither is specified, it relies on a verse difficulty metric that has been precomputed using a dedicated algorithm. The service then navigates through the verses in the appropriate direction, starting from the verse immediately after the last one recited, and accumulates the required amount—whether in letters, pages, or difficulty, until it reaches the target suitable for the student for that day. If the student is close to finishing a surah and only a small portion remains, the plan is adjusted to either stop at the end of the surah or to complete the surah if the remaining amount is within a certain threshold, ensuring a logical and realistic progression that aligns with both practical teaching considerations and AI recommendations.

For major revision, the service employs the same core logic but adapts dynamically to the student's memorization and revision direction. If the student's revision direction is set to descending (from Al-Fatiha to An-Nas), the service progresses through the Quran in descending order. Upon reaching An-Nas, it automatically resets to either the surah immediately preceding the student's current memorization point (if memorizing in descending order) or the surah immediately following (if memorizing in ascending order). For students revising in ascending order (from An-Nas to Al-Fatiha). This keeps the revision aligned with what the student is currently memorizing while covering the whole

Quran. It also avoids mixing revision with new memorization, whether moving forward or backward through the Quran.

For minor revision, the process is slightly different: the system reviews the student's previous new memorization sessions prior to the current day, sequentially adding them until the total amount matches the student's specified capacity, whether measured in letters or pages. This ensures that the minor revision plan is closely tied to the student's actual memorization history and abilities. This process is repeated for each of the selected days in the week, with the resulting daily plans being stored in the database. By dynamically adapting to the student's progress, preferences, and historical performance, the Plan Generation Service provides a structured yet flexible roadmap for Quranic memorization and revision, ensuring that each student receives a plan that is both achievable and tailored to their individual needs.

5.5.3.6. Manage Recitation Evaluation Service

The Manage Recitation Evaluation Service class is a crucial part of the backend system, responsible for handling the evaluation and adjustment of students' recitation sessions. This service oversees the process of recording and updating the results of each recitation session, including whether the student's performance was accepted, the grade or rating assigned, and any changes to the range of verses covered. When a session is evaluated, the service recalculates the student's overall and section-specific ratings, ensuring that these metrics accurately reflect the student's most recent performance. If a session is not accepted, the service intelligently adjusts subsequent sessions to maintain consistency in the student's memorization and revision plan, either by shifting the range of verses or, in some cases, removing sessions that are no longer relevant. The service also monitors for any changes in the verses recited, and if such changes occur, it can trigger the regeneration of future plans to ensure that the student's schedule remains aligned with their actual progress. By managing these complex interactions between session evaluation, ratings, and plan updates, the Manage Recitation Evaluation Service ensures that the system remains adaptive and responsive to each student's needs, providing a robust framework for continuous improvement and personalized learning.

5.5.4. Database Models and ORM Integration

The data layer is defined by SQLAlchemy models that map to database tables, including students, surahs, verses, and recitation_sessions. These models enforce data integrity through validation constraints and define relationships (e.g., linking students to their plans or sessions). By abstracting SQL operations into Python objects, the models enable services to manipulate data intuitively while maintaining separation between business logic and database structure. This approach simplifies complex queries, supports cascading updates, and ensures scalability.

In summary, the backend architecture successfully balances adaptability, scalability, and robustness, providing a foundation for personalized Quran memorization management. By decoupling components into modular services, route handlers, and models, the system ensures maintainability and clarity. The integration of AI-driven recommendations, bidirectional verse navigation, and dynamic plan adjustments reflects a commitment to pedagogical effectiveness. Combined with rigorous error handling and transactional integrity, the backend not only meets current demands but also provides a flexible framework for future enhancements, ultimately supporting learners in achieving their memorization goals efficiently.

CHAPTER 6

TESTING

6.1. LSTM Models Testing

The testing phase evaluates the performance of LSTM models trained for three types of memorization activities: new memorization (Pillar 1), minor revision (Pillar 2), and major revision (Pillar 3). The models analyze sequences of historical student sessions to predict the workload for subsequent sessions. For example, if a student memorized 700 letters in their last nine sessions on average, the model predicts how many letters they should aim for in the next session. These predictions are used to generate structured memorization plans, specifying verse ranges, total letters, and Surah IDs.

The results of the testing phase reveal varying levels of accuracy across the three pillars. For new memorization (Pillar 1), the model explains approximately 47% of the variance in predictions, with an average error of 174 letters. While this indicates some ability to recognize patterns, moderate accuracy suggests limitations in capturing the full complexity of student learning behaviors. Minor revision (Pillar 2) shows stronger performance, with the model explaining 70% of the variance, though the average error increases to 300 letters due to the naturally variable nature of revision tasks. For the major revision (Pillar 3), the model explains 64% of the variance and has the highest average error (1,265 letters), reflecting the challenges of predicting long-term retention efforts.

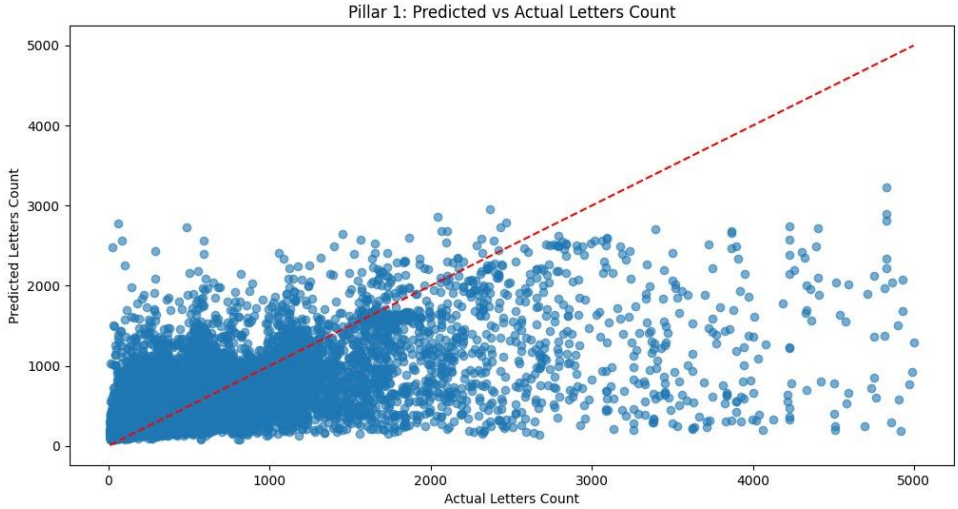

Figure 22: Results for New Memorization

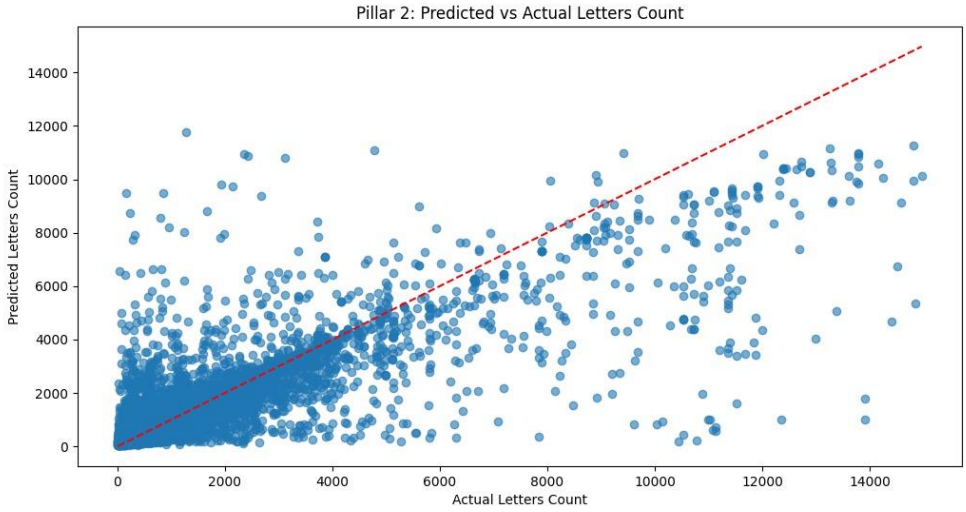

Figure 23: Results for Minor Revision

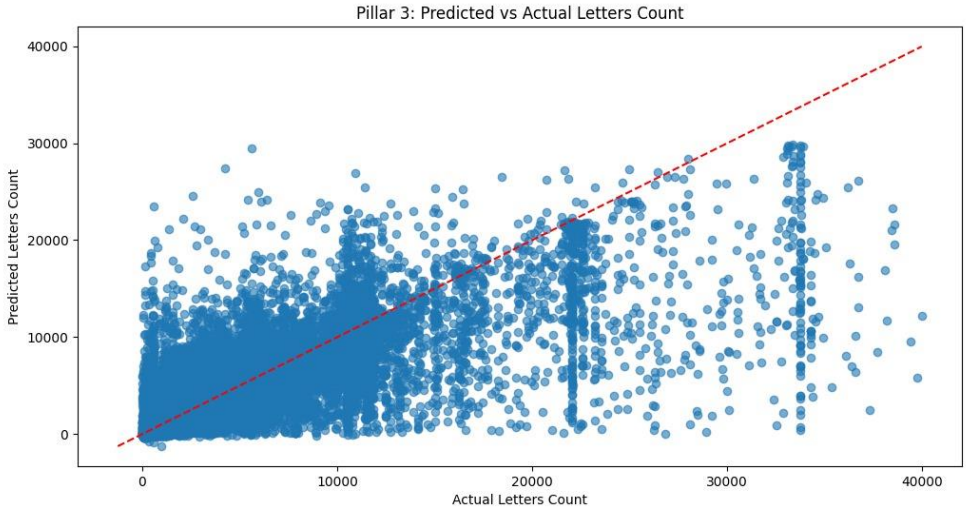

Figure 24: Results for Minor Revision

The testing results partially reflect the model's ability, demonstrating moderate success in identifying sequential patterns but revealing significant limitations, such as high prediction errors and an inability to account for real-world variables like student motivation or sudden learning plateaus. While the models provide a foundation for generating baseline memorization plans, their current performance, rooted in historical data under controlled conditions, does not fully capture practical effectiveness, as they lack dynamic adaptability and validation against real-world outcomes. To realize their potential, the models require real-world testing to ensure they meet educators' and students' needs in precision and adaptability.

6.2. System Testing

A robust and reliable system requires a comprehensive approach to testing that encompasses not only backend functionality but also the seamless integration with frontend components and other system modules. The system testing was designed to validate the end-to-end workflow, ensuring that each layer—from the database and

business logic to the user interface—operates correctly both in isolation and as part of the complete application.

The testing process began with a focus on the backend, where we implemented a suite of automated tests targeting the most critical API endpoints. These endpoints were chosen for their central role in delivering core data to users, including student records and recitation session histories. The tests confirmed not only that the routes were accessible and correctly registered within the application context, but also that they returned consistent and accurate responses under various conditions. This initial phase of testing provided confidence that the backend infrastructure was stable and that the fundamental data retrieval mechanisms were functioning as intended.

Beyond route-level validation, we extended our testing efforts to the backend's service layer, which encapsulates the application's business logic. Here, we examined the modules responsible for generating personalized memorization plans, managing session data, and handling user-specific operations. By simulating a range of scenarios—including different student profiles, progression stages, and edge cases—we ensured that the logic was robust and adaptable. These tests were instrumental in identifying and resolving subtle issues that could arise from complex user interactions or atypical data states, thereby strengthening the reliability of the backend before it was exposed to real-world usage.

Integration testing formed the next critical phase, bridging the backend with the frontend interface. After confirming the correctness of the API endpoints, we connected them to the frontend components, such as dashboards, profile pages, and weekly plan displays. This integration was tested by navigating through the user interface and observing the live data exchanges between the frontend and backend. We closely monitored the flow of requests and responses, paying particular attention to data consistency, error handling, and the clarity of user feedback. Any discrepancies or unexpected behaviors were promptly addressed, either by refining the backend logic or by adjusting the frontend's data handling routines.

Through this layered approach to system testing, beginning with isolated backend verification, advancing comprehensive service logic checks, and culminating in full-stack integration validation, we achieved a high degree of confidence in the system's overall quality. The successful interaction between the backend and frontend not only demonstrates the technical soundness of our implementation but also highlights the effectiveness of our testing methodology. This thorough process ensures that the system delivers a reliable and user-friendly experience and is well-prepared to handle the demands of real-world deployment.

CHAPTER 7

FUTURE WORK AND CONCLUSION

7.1. Future Work

The proposed reinforcement learning-based Quran memorization system presents several promising avenues for future enhancement and expansion. A critical next step involves deploying the model as a scalable web service or mobile application to broaden accessibility, particularly for large-scale Quranic learning circles and diverse community segments. This transition would necessitate addressing practical deployment challenges, such as optimizing computational efficiency and ensuring robust cross-platform compatibility. Additionally, expanding the system's functionality to serve multi-tier user roles, including administrators, teachers, parents, and students, could foster a more collaborative and institutionally integrated ecosystem.

To further personalize learning experiences, advanced AI-driven features such as automated voice recognition for recitation evaluation could replace manual teacher assessments, enhancing objectivity and scalability. Integrating comprehensive student performance analytics, including automated monthly progress summaries in PDF format, would empower stakeholders with actionable insights. A centralized web-based dashboard could be developed to monitor classrooms, manage users, and export data, while mobile app notifications could deliver real-time reminders for tasks, deadlines, and progress updates.

Future iterations should also prioritize refining the reward function through diverse and representative training datasets, capturing varied recitation styles and learner demographics, to improve adaptability across cultural and linguistic contexts. Collectively, these enhancements would strengthen the system's educational impact, scalability, and real-world applicability, positioning it as a transformative tool for Quranic education initiatives.

7.2. Conclusion

The development of an AI-driven Quran Memorization Plan Generator marks a significant advancement in personalized Quran education. By integrating Long Short-Term Memory (LSTM) networks and Reinforcement Learning (RL), the system addresses the strictness of traditional memorization methods, offering dynamic, adaptive plans tailored to individual student performance. The LSTM model effectively identifies sequential patterns in historical memorization data, while the RL component enables real-time adjustments based on recitation feedback, ensuring a balance between progress and retention.

The database architecture, with its interconnected tables for users, students, verses, surahs, and recitation sessions, provides a robust foundation for tracking and analyzing learning trajectories. Testing demonstrated moderate predictive accuracy, with the LSTM explaining up to 70% of variance in revision tasks, though higher error rates in major revisions highlight opportunities for model refinement. The backend services and frontend interface further ensure seamless interaction, enabling educators to manage plans, evaluate progress, and adapt strategies efficiently.

This project underscores the transformative potential of AI in Quranic education, offering scalable solutions to enhance memorization efficiency and long-term retention. Future work should focus on deploying the system as a web or mobile application, integrating automated voice recognition for recitation evaluation, and expanding user roles to accommodate administrators, teachers, and parents. By addressing current limitations and leveraging advanced AI capabilities, the system can evolve into a comprehensive tool that bridges tradition with technology, empowering learners worldwide to achieve their Quranic memorization goals with confidence and adaptability.

References

[1] K. Haryono, R. A. Rajagede, and M. Negara, "Quran Memorization Technologies and Methods: Literature Review," *IJID (International Journal on Informatics for Development)*, vol. 11, pp. 192–201, Jan. 2023, doi: 10.14421/ijid.2022.3746.

[2] S. Maghsudi, A. Lan, J. Xu, and M. van der Schaar, "Personalized Education in the Artificial Intelligence Era: What to Expect Next," *IEEE Signal Processing Magazine*, vol. 38, no. 3, pp. 37–50, May 2021, doi: 10.1109/MSP.2021.3055032.

[3] Research Scholar from School of Computer and Information Sciences, University of Hyderabad.., G. R. Anil, S. A. Moiz, and Assistant Professor at School of Computer and Information Sciences, University of Hyderabad., "Personalized Dynamic Learning Plan Generator for Smart Learning Environments," *IJRTE*, vol. 8, no. 2, pp. 6175–6180, Jul. 2019, doi: 10.35940/ijrte.B3806.078219.

[4] M. Somasundaram, K. A. M. Junaid, and S. Mangadu, "Artificial Intelligence (AI) Enabled Intelligent Quality Management System (IQMS) For Personalized Learning Path," *Procedia Computer Science*, vol. 172, pp. 438–442, Jan. 2020, doi: 10.1016/j.procs.2020.05.096.

[5] H. Oussama, N. El Faddouli, M. H. Alaoui Harouni, and J. Lu, "Artificial Intelligent in Education," *Sustainability*, vol. Volume 14, p. 5: 2862, Mar. 2022, doi: 10.3390/su14052862.

[6] L. Zhang, J. D. Basham, and S. Yang, "Understanding the implementation of personalized learning: A research synthesis," *Educational Research Review*, vol. 31, p. 100339, Nov. 2020, doi: 10.1016/j.edurev.2020.100339.

[7] D. Niu, Z. Xia, Y. Liu, T. Cai, T. Liu, and Y. Zhan, "ALSTM: Adaptive LSTM for Durative Sequential Data," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov. 2018, pp. 151–157. doi: 10.1109/ICTAI.2018.00032.

[8] R. S. Sutton and A. Barto, *Reinforcement learning: an introduction*, Second edition. in Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, 2020.

[9] "Time series forecasting | TensorFlow Core," TensorFlow. Accessed: Mar. 09, 2025. [Online]. Available: https://www.tensorflow.org/tutorials/structured_data/time_series

[10] J. Brownlee, "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras," MachineLearningMastery.com. Accessed: Mar. 09, 2025. [Online]. Available: https://www.machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

[11] "PPO — Stable Baselines3 2.6.1a0 documentation." Accessed: Apr. 16, 2025. [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

[12] *DLR-RM/stable-baselines3*. (Apr. 15, 2025). Python. DLR-RM. Accessed: Apr. 16, 2025. [Online]. Available: https://github.com/DLR-RM/stable-baselines3

[13] "Gym Documentation." Accessed: Apr. 16, 2025. [Online]. Available: https://www.gymlibrary.dev/
[14] "SQLAlchemy." Accessed: Apr. 16, 2025. [Online]. Available: https://www.sqlalchemy.org
[15] M. O. contributors Jacob Thornton, and Bootstrap, "Introduction." Accessed: Mar. 09, 2025. [Online]. Available: https://getbootstrap.com/docs/4.1/getting-started/introduction/